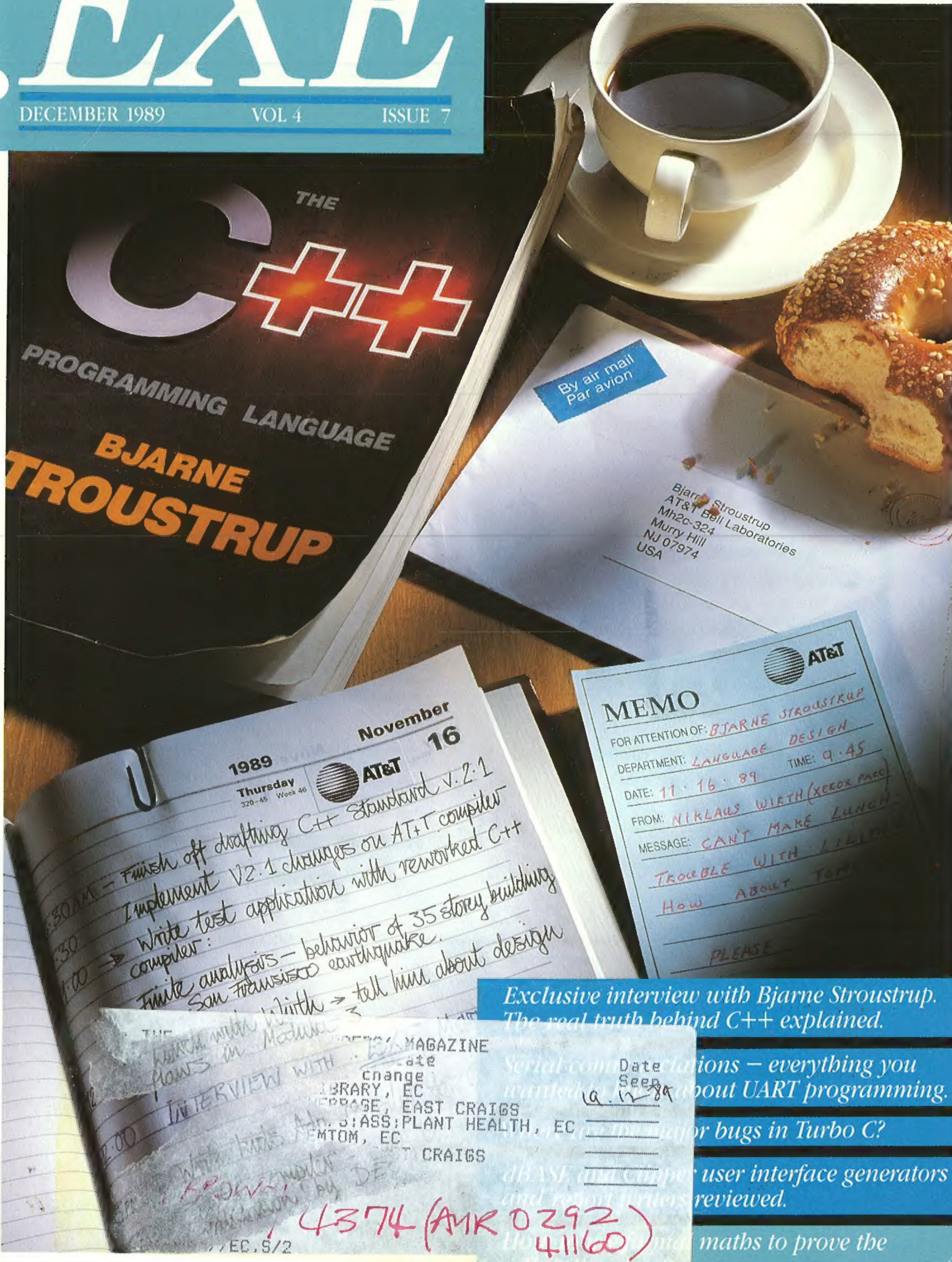
.EXE

£3.20

The Software Developers' Magazine

DECEMBER 1989 VOL 4 ISSUE 7



Exclusive interview with Bjarne Stroustrup. The real truth behind C++ explained.

Serial communications — everything you wanted to know about UART programming.

Where are the major bugs in Turbo C?

dBASE and Clipper user interface generators and report writers reviewed.

How [illegible] formal maths to prove the reliability of software.

THE TRAINING LEADERS

FOR LONDON BASED OR ON-SITE TRAINING

"You don't have to come to our castle to crown your software skills"

QA Training is the acknowledged leader in technical PC training.

We are noted for the quality of our courses, the depth of experience of our instructors, the provision of very professional computer work stations – and for our castle, Britain's most advanced PC training facility.

But if you need all the benefits of a QA training course and find London a more convenient venue ... we offer a range of London based courses in the areas of OS/2, Languages and System Support. Our London courses maintain the very high standards that QA has become known for.

And if London isn't quite right we are happy to provide on-site training at your own castle.

Send for more details today or call Samantha Trinder on 0285 655888.

LONDON COURSES

OS/2 Programming – 4 Days
6 February, 3 April

OS/2 LAN Programming – 4 Days, 17 April

OS/2 Presentation Manager Programming – 5 Days
29 January, 19 February, 2 April

Microsoft Windows Programming – 4 Days
23 January, 17 April

Commonview – 1 Day
16 March

DOS Application Programming – 4 Days
27 February

C Primer – 2 Days
4 January, 30 January, 27 March

C Programming – 4 Days
8 January, 22 January, 5 February, 12 February, 5 March, 12 March, 26 March, 9 April, 23 April

C Techniques OS/2 and PM – 1 Day
27 April

C Techniques for DOS – 1 Day
16 February, 16 March

C Advanced Software Development – 4 Days
20 February, 20 March

C Standards – 1 Day
30 March

C++ Programming – 4 Days
12 March

Pascal Programming – 4 Days
13 February

DOS Primer – 2 Days
1 February, 29 March

Supporting Personal Computers – 4 Days
9 January, 6 March, 9 April, 24 April

Supporting OS/2 – 4 Days
16 January

Supporting OS/2 LAN Manager – 4 Days
27 February

Supporting IBM DOS and OS/2 LANS – 4 Days
20 March

OS/2 Strategic Briefing – 1 Day
29 January, 19 March

OS/2 Technical Overview – 1 Day
5 February, 26 March

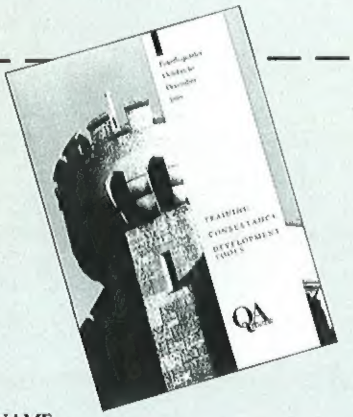
OS/2 Networking – 1 Day
26 February, 23 April

OS/2 Presentation Manager – 1 Day
12 February, 2 April

OS/2 SQL Databases – 1 Day
26 January, 19 March

Microsoft Authorised

IBM Suppliers of OS/2 training and education materials to IBM



Please complete and mail to QA Training, Cecily Hill Castle, Cirencester, Glos GL7 2EF

☐ Tick here for full details on LONDON courses

I am also interested in QA courses on the following (tick as appropriate):
☐ LANGUAGES ☐ OS/2 ☐ WINDOWS ☐ DOS ☐ SUPPORT

NAME

TITLE

COMPANY

ADDRESS

TEL. NO.

QA TRAINING

CIRCLE NO. 696

Vol 4 Issue 7

.EXE

December 1989

Editor
Robert Schifreen

News Editor
Will Watts

Editorial Assistant
Wendie Leek

Advertising Manager
Philip Northcott

Sales Executives
Sandra Inniss-Palmer
Jeremy Cox

Publisher
Jon Howell

Reader Services & Promotions
Helena Adams

Design & Layout
Katerina Adams
Rachel Affleck

Office Administrator
Carla Pearce

Cover Photography
Richard Cook

Editorial Policy
We aim to provide news, product reviews and technical features for those who develop PC software for both commercial sale and internal company use. Our policy is not to review any software product until it is available in its final form, in order to provide accurate figures on code size and speed. The Magazine welcomes articles from readers – please ask for our contributor's guide.

Subscriptions
.EXE is a magazine for managers and programmers involved in the development of software for the IBM PC and compatibles. .EXE is available only on subscription, which costs £35 per year (11 issues) in the UK. The Magazine is published on the 26th of the month preceding the cover date. There is no January issue. International subscription rates can be found on the subscription card in this issue. Back issues are available at £3.50 each. To subscribe, call 044282 4501. We can invoice your company or take VISA or ACCESS cards.

.EXE Magazine is independent and not affiliated to any vendor of hardware, software, or services. Published by Process Communications Ltd. 10, Barley Mow Passage, London W4 4PH, England. Telephone 01 994 6477

ISSN 0268-6872

Reacting to Readers
We aim to provide high quality facts and information, about subjects relevant to readers. Our annual Reader Survey gives us an insight into the tools, methods language and hardware that is being used, and that will be used, in the process of software development.

We aim to keep in touch with software developers as much as possible. For this reason, we hold twice-yearly meetings at the .EXE offices and around the country, where readers can talk directly to the editorial staff and suggest ways in which the Magazine could be improved.

Wot, no Virus?

Viruses and hacking both hit the headlines recently, and the coverage was not confined to the computer press alone. The Law Commission published a long-awaited report that suggested how an anti-hacking Bill could be drafted. Around the same time came Friday 13th, which was the activation day of a virus called Datacrime, that was supposed to bring every computer in the world to its knees and wipe out every disk in the land.

A lot has been said on both subjects, and much of what was said was untrue. Some was untrue because it was written and reported by journalists who didn't really understand computers, but who thought that they did. Some was untrue because it appeared in the tabloids.

We don't normally devote space in .EXE to the subject of viruses, because they really are not as widespread as the national press would have us believe. Only around 4 or 5 cases a day are actually discovered in the UK which, considering the number of PC users and the huge amount of software copying that goes on, is not quite an epidemic. I certainly don't condone the planting of viruses. I know people that have been the subject of a virus attack, and I know the huge amount of inconvenience that Mac users are going through all over the world, but the number of PC users who have actually lost data is very small.

Britain has its fair share of virus experts, all of whom were approached many times by television news teams, and asked for a quote on just how painfully the world would end on Friday 13th. These experts told the quote-seekers the truth; that there were literally no reported cases of the Datacrime virus in the UK, and that there would be no mass outbreak of corrupted floppies. Quotes of this kind aren't considered newsworthy, so the experts did not make it as far as the gates of TV Centre.

In the end, it was the Royal National Institute for the Blind that made it to the TV screens. Viewers were introduced to a very sad programmer at the RNIB computer centre, who has spent the last 4 years designing and writing specialist software to help blind and partially sighted people use PCs. He had, it seemed, lost all his 4 years' of work.

After a week or so, it emerged that, as many had suspected, the RNIB had not contracted the Datacrime virus at all. The virus was one known as 1813, which is an irritant but not a disaster. I have it from the person employed to clean out their computer centre that, within a couple of hours, their entire collection of hard and floppy disks had been restored to full health, and not a single byte of data had been lost. I have yet to see this fact reported on the TV.

As for hacking, the Law Commission suggests that three new offences be created. Hacking for fun would get you up to three months. Hacking with the intention of gaining money or services would get you five years, as would actually succeeding in getting any of the above. My personal opinion is that any Act drawn up according to these guidelines would not provide much benefit.

In my opinion, extra legislation is not what is required. We need to educate the computer-using public that, if you leave doors unlocked, someone is bound to come along and try them. If you're lucky, the person who gets in will call you and warn you that you've left the door unlocked. With the threat of a jail sentence hanging over his head, though, perhaps he may not.

GREY MATTER

4 Prigg Meadow, Ashburton, Devon TQ13 7DF.
TEL. (0364) 53499

GREY MATTER

4 Prigg Meadow, Ashburton, Devon TQ13 7DF.
TEL. (0364) 53499

Power Phoenix from Turbo Ashes

As reported in last month's .EXE, Borland has abandoned development of Turbo BASIC V2.0. The good news for frustrated users of Turbo BASIC V1.1 (see Letters page of recent months) is that the product lives on, although no longer under Borland's aegis.

Borland has surrendered the development and publishing rights to the program's original developer, Robert Zale. Mr Zale, in turn, has reached agreement with the Californian company Spectra Software to publish what would have been Turbo BASIC V2.0 as PowerBASIC. This product is supposed to ship in December in the US priced about $110. There is even an upgrade policy for registered users of Turbo BASIC V1.1, priced at $50. Details concerning the availability of PowerBASIC in the UK had not been finalised at time of going to press. We will tell you more when we know.

Compaq goes Multi-processor

There seem to have been dozens of new machines announced over the past month, as the EISA 32-bit bus standard – the two fingers waved at MCA standard – finally begins to bear fruit. Of all these, Compaq's Systempro walks off with the prize for outrageously OTT specification. Take a look at this: 33 MHz 80386 with 64 KB of cache memory, 32-bit EISA bus, 4 MB of 32-bit memory expandable to 256 MB, an Intelligent Drive Array controller (based on a 16 MHz 80186) equipped with up to 840 MB of hard disk (with capacity for 1.68 gigabytes within the machine) and, wait for it, the space for a second CPU.

Actually, this is a slight simplification. There are four processor board slots: two for 386 cards and two for 33 MHz 486s (when available). You can populate the machine with any combination of CPU types, but you are only allowed two processors. This may be enough: according to Compaq, the Systempro ran the Neal Nelson benchmark six times as fast as a VAX 6310 – a machine priced at just $135,000 more.

What will you run on it? Novell Netware 386, Compaq LAN Manager 386/486 or an enhanced SCO UNIX System V. This last, which will ship early next year, is based on the standard SCO kernel with a special multi-processor extension. Each processor executes system and user code as though it had the whole machine to itself. The uniprocessor scheduler has been extended so that the highest priority system task is executed by the next free CPU – so the processing load is kept in balance. The starting price for the Systempro range is around £11,000; shipments should begin this month.

Amstrad BASIC

Locomotive Software, creators of the successful LocoScript word processor supplied with the Amstrad CP/M machines, has upgraded its BASIC product. BASIC 2 Plus, which runs on 1512 and 1640 machines, is supposed to run up to six times as fast as its predecessor. Other enhancements include a WIMP editor, improved arithmetic precision and GEM 3 compatibility. The program costs £49.95 ex VAT and is available direct from its makers (phone 0306 740606).

OS/2 Update

At Comdex, the recent American computer show, Microsoft and IBM issued a number of joint statements. OS/2 2.0 (the 386 version) will ship to developers in the US before the end of the year, and to end users sometime during 1990. No date for UK shipments of either package was given. The IBM and Microsoft version of the product will be identical, which is not currently the case. Base memory requirement will double, to 4 MB. There will be support for multiple processors (this despite previous denials from Microsoft; I know someone who has seen OS/2 running with an i860 as a graphics processor) and enhanced security features. Incidentally, recent US rumours have suggested that IBM's Extended Edition of OS/2 1.2 has been delayed until March 1990, with LAN problems being the suggested cause.

Recent press speculation has hinted that IBM's UK research labs have developed a cut-down version of Presentation Manager, which IBM hopes to sell as a direct competitor to Microsoft's Windows. It was said that the product would be called 'PM Lite', following the convention adopted by the makers of low-calorie foods in the US. IBM and Microsoft stated publicly at Comdex that this product does not exist. There seems little reason to believe that it ever will. IBM and Microsoft developed OS/2 under a joint development agreement. It would be most surprising if the contract permitted IBM to market a product derived from PM without Microsoft's consent.

Release 4

UNIX International and AT&T have finally introduced UNIX System V Release 4. This long-awaited product is supposed to twist the disparate strands of the operating system (Xenix, SunOS, BSD 4.2 and 4.3) into one all-compatible golden thread. However, at least one thick blue strand, IBM's AIX, remains conspicuously aloof. It is not yet clear who will regret this.

New Sidekick

Borland has launched Sidekick V2.0, for PM priced at £199.95. Like the previous version, the application consists of a diary, an editor, a phone book and a calculator; the main enhancement is an interface to the unreleased Paradox engine. The program requires OS/2 V1.2 to run.

Lattice secrets

Lattice Inc, of C compiler fame, has produced SecretDisk II, an MS-DOS product which automatically encrypts data as it is written to disk. Access to the data is obtained by giving a password. Lattice suggests that the $125 program could be used on lap-top computers. If the machine is stolen, the data on it is useless to the thief. Call Lattice in the US on 0101 708 916 1600, or Lattice C UK distributor Roundhill may have it (0672 84535).

DESQview Distributed

For a long time, there was no official UK distributor for DESQview, the multitasker for DOS programs. Now, we hear, International Data Security are the people you need to speak to. Their phone number is 01-631 0548.

LISP Conference

EUROPAL '90 is the first European conference on the practical application of LISP. It is to be held in Cambridge during March 1990. The intention of the conference is to improve awareness of the capabilities of LISP; particular attention will be paid to its suitability for developing object-oriented systems. If you would like further details of the conference or a copy of the Call for Papers, contact David Lloyd on 0306 889485.

Ada Graphics

Media Cybernetics has produced a HALO graphics toolkit with PC Ada language bindings. HALO Ada will drive 110 graphics devices, including various extended VGA boards and the IBM 8514A. The package supports the Alsys, Meridian and Janus compilers. It costs around $1000 and is available direct from the US. Call 0101 301 495 3305 for details.

Multi-user OS/2

IBM has two operating systems for the PS/2 machines, namely OS/2 and AIX. The Company's marketing departments list 'multi-user capabilities' as the primary difference between the two. Those who want multiple users are told that they need to go down the AIX route; those who want to hog a machine's power to themselves are sold OS/2. At the recent OS/2 Show, Qiiq Ltd upset the apple cart with Terminal Manager. On their stand, a PS/2, running OS/2, was hooked up to a couple of Wyse 60 terminals. These behaved like the console of an OS/2 V1.0 machine. The program selector worked, as did character-based software.

Terminal Manager V1.0 is promised in December. The program consists of a system monitor that takes up just one screen group on a standard machine. This monitor sits between the OS/2 kernel and the I/O drivers, intercepting screen-bound characters and sending them to a serial port. Incoming characters are gathered as they appear at a comms port, and routed to the correct application. Because badly behaved protected mode programs are rare (there is no writing to hardware ports, for example), the monitor seems to catch all keystrokes and characters. The only problem that Qiiq had was in OS/2's serial port drivers. In the end, they wrote their own, which handle speeds of up to 9600 bps quite comfortably.

The next release of TM is scheduled for next May, by which time the product will support dial-up lines and a larger variety of terminals (it's just VT100 at the moment). There will also be the ability to restrict users to special shells, and to introduce a measure of security into the system. At present, all users execute the standard CMD.EXE command interpreter. The product supports a number of standard multi-port cards, and typical prices are £595 for a 16 user system, or £395 for eight users. That's £37 and £49 per user, respectively. More information from Qiiq on 01 541 0388.

Danes turn to Piracy

Link Computer, a Danish Software House, is launching its MS-DOS copy protection software program COP'S COPYLOCK II in the UK. The product has already been on the European market for over four years; the company says that the latest version has remained uncracked for one and a half years.

One of CopyLock's claimed unique features is that it takes a 'fingerprint' of the floppy or hard disk that contains the protected program file. You can copy around the program, but it will only work on the disk where it has been installed.

Software protected by CopyLock is safe from bit copying programs such as Copy II PC and CopyWrite. The manufacturer says that, unlike the similar Everlock, CopyLock is secure from hardware copying devices. Dino Cano, Link's UK Marketing Manager, told me: 'Copy Protection has been associated with installation and backup problems, but these were simply technical limitations that CopyLock does not have.' We have yet to see for ourselves. The package costs £1200. For further information contact Link Computer in Denmark on 01045 3123 2350.

Nife

Have you ever used EDLIN, run from a batch file with a series of keystrokes piped in from a pre-prepared text file, to correct some vast and widespread error that is repeated hundreds of times throughout many files (eg substituting decimal point for decimal comma in files of Continental origin)? The supplementary question: didn't you wish there was a better tool than EDLIN to do the job?

NIFE is the better tool. It is a rare beast: an MS-DOS editor designed to be run in batch mode. You give it a source file, a file of commands and the name of your target; you then let it alone to do the work. The core of the program is NIFE's instruction set. As well as the obvious stuff like Search and Replace you can centre lines, mark blocks, test conditions (eg is the sixth word in this line 'hello'?), combine the tests with standard and/or/not logic and generally have a ball. Use it to convert between word processing formats, to enforce company standards for documents and program code and to insert typesetting codes automatically.

I have a feeling that this is one of those programs whose use is not immediately obvious, but, once the point has been grasped, turns out to be invaluable. The developer is Bristol-based Cadspa Ltd – call them on 0272 326768. Nife costs £95 ex VAT.

VAX Editor

If you have made the switch from VAXs to PCs, you may regret having lost access to the superb VMS text editors. Software distributor Ethan Adams (0530 412405) has released a product called nu/TPU, which emulates the VMS Text Processing Utility under MS-DOS and certain flavours of UNIX. The program is claimed to have all the functionality of the original, including user definable interfaces, an embedded procedural language and EDT and EVE editing key-pads. Prices start at £275.

DR Disks

Digital Research, of GEM and CDOS fame, has announced a range of services that it is offering to other software houses. These include floppy disk sales and duplication, printing, packaging and assembly and warehousing. Only a cad would speculate that all this spare capacity suggests that DR's own products are not selling so well. Details from James Harbin at DR's Hungerford office on 0488 84587.

Superior GKS

S-GKS 2.1 is a Dutch implementation of the ISO GKS graphics standard, with various extras such as bit map operations and three types of metafile. The system runs on a range of workstations (including Sun and Apollo) as well as the MS-DOS based PC. There is support for Postscript printers. Prices for a developer's licence start at £550 for the PC version up to £1600 for workstation software. Call Scientific Software Ltd on 0628 890011.

Objectworks for C++

Objectworks is a software development system for AT&T's C++ Release 2. It consists of three tools: an incremental compiler (which actually calls the C++ translator and the host C compiler), source-level debugging and source code browsing. It currently only runs on the Sun-3 – other platforms to follow – and costs £2200. Contact the UK distributor AI Ltd on 0923 247707.

386 to the max

386MAX is one of those memory management utilities that lets you access 'spare' memory between the 640 KB MS-DOS limit and the 1MB real mode ceiling. By moving network drivers into this space, it increases the amount of RAM available to network applications. The program itself uses 64 bytes of conventional memory. It costs £89.95. Phone First National Sales Ltd on 0491 25555 for more details.

IBM AIX PS/2. Greater price perfo

rmance than you thought possible.

The remarkable AIX PS/2. A multi-user operating system that delivers the highest performance at a more than competitive price. A version of IBM's AIX specifically designed to run on the most sophisticated PCs in the world, the top of the range IBM PS/2s.

Compare AIX PS/2*with the best of the competition:

Performance. An operating system optimised to run on IBM Micro Channel* Architecture with true 32-bit processing, delivering faster and more efficient performance for multi-user applications.

Price/performance. Particularly impressive thanks to this optimisation of hardware and software at personal computer prices.

Functionality. IBM has enhanced the UNIX# operating system. There are major improvements, especially in ease of use and installation, security and comprehensive communications.

Applications. Thousands of UNIX applications can easily be ported to AIX. Many are already available and the list is growing rapidly.

Current investment. As AIX PS/2 and DOS sessions can run concurrently, your existing investment in DOS applications is protected.

Growth potential. AIX gives you the widest range of UNIX hardware platforms: from PS/2 right up to IBM mainframes. A capacity which should absorb even the most spectacular growth.

AIX and PS/2. A sterling partnership worth thinking about from IBM.

For more information, contact Bruce Young at the National Enquiry Centre, IBM UK Ltd, FREEPOST, 389 Chiswick High Road, London W4 5BR or telephone 0800 181182 during working hours. "I think, therefore IBM."

IBM

CIRCLE NO. 698

*AIX, Micro Channel and PS/2 are trademarks of International Business Machines Corporation.
#UNIX is a trademark of AT&T in the USA and other countries.

FORTRAN Fun

Two issues ago we reported the knots into which ANSI had tied itself over the C standard. These are but nothing compared to the activities of the various FORTRAN committees, and in particular those of X3J3, the technical committee charged with processing the draft FORTRAN 8X standard, the intended successor to FORTRAN 77.

To explain: FORTRAN 8X is a radical revision of its predecessor. For the last two years someone in the FORTRAN community, rumoured to be a compiler vendor, has been running an 'unprincipled campaign' to prevent the new standard's acceptance. This unknown party is supposed to have encouraged members of the public, who had not even read the proposals, to register comments on the draft standard. One has images of a man standing in the street with a clipboard, assaulting passers-by with 'Excuse me, sir; have you, as a good citizen, exercised your Democratic Right to object to the new FORTRAN standard?'

X3J3 members who complain about this behaviour are dubbed 'un-American' (the worst insult possible in the US, without resorting to filthy words like 'Communist'); in fact the deliberations of the previously stiff-necked committee are beginning to look like a transcript of *Julian Clary's Sticky Moments*.

A side effect of this is that the possibility now exists that ANSI and the International Standards Organisation (which has not been infiltrated, and of which our own BSI is a member) will produce two different FORTRAN standards: the radical ISO FORTRAN 8X and the conservative ANSI 'FORTRAN 77.01'. It's lucky that we have committees to sort these things out, isn't it?

Dyadic's APL

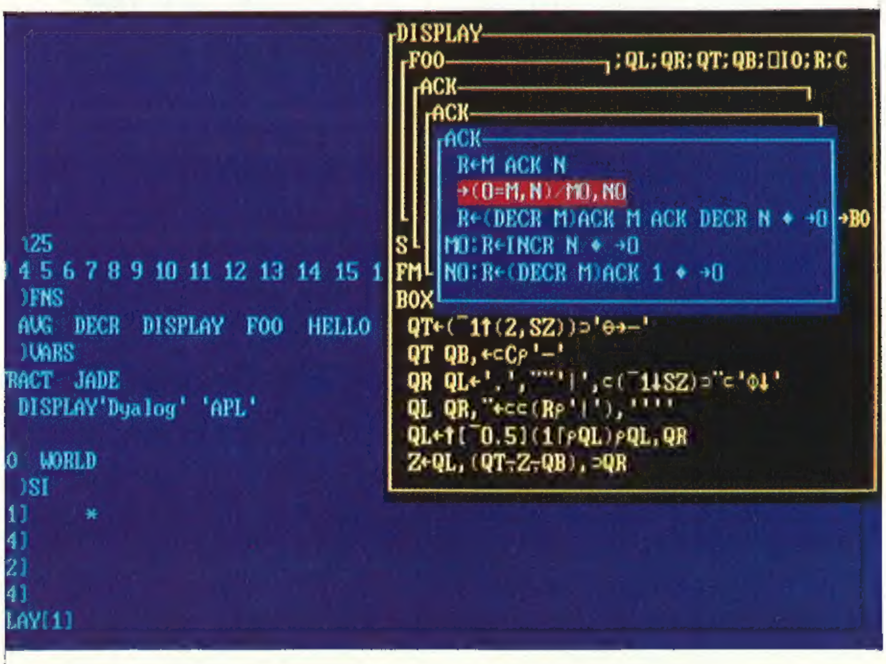
Hampshire company Dyadic has announced a new version of its Dyalog APL product for MS-DOS 386 PCs. Dyalog APL is a second generation APL, incorporating such features as nested arrays, the 'each' operator, user-defined operators and selective specification. The package includes a sophisticated development environment, which permits multiple edit windows and also has a source level debugger.

The manufacturers are particularly proud of their method of providing screen access. This is achieved via a system variable called ⎕SM. Changes to ⎕SM are immediately reflected by an update to the display. For example:

```
⎕SM[2;4 5 6]← 24 80, RED ON BLUE
```

blows up field 2 to screen size (24 x 80) and alters its colour to red on blue. Pop-up windows are neatly supported by localising ⎕SM.

Those odd-looking APL characters are created in text mode using EGA definable fonts, so you will need an EGA or VGA card in your machine. You will also need at least 2 MB of RAM (the software was written using Metaware's High C, and runs in 32-bit native mode) and an 80387 coprocessor is recommended. Dyalog APL for DOS costs £995. Call Dyadic on 0420 87024.



486 High C

Metaware has announced V1.6 of its High C compiler for 80386/80486-based PCs. The compiler can generate 32-bit protected mode object code which runs with MS-DOS extenders (such as Phar Lap's) and, for the first time in any compiler that we have heard of in this office, provides specific support for the 486. Other new features include a protected mode graphics library and additions to the base library to make it '86% compatible' with Microsoft C's libraries. The company is simultaneously releasing OS/2 and MS-DOS versions of High C. More information direct from Metaware in the US on 0101 408 429 6382.

Algebra

'If it's so clever', I sneered at my maths teacher when he showed us the school computer for the first time, 'how come it doesn't do calculus?' The Derive program, for MS-DOS PCs, not only does derivations; it also does formula simplifications, quadratics, matrices, trigonometry and complex numbers. Just the thing if you are constantly struggling with someone else's A level Maths. Derive costs £125 from Chartwell-Bratt (01 467 1956).

IMS on a PC

IMS programmers can now develop COBOL applications without involving the mainframe. Realia has produced RealIMS, a compatible MS-DOS based environment designed to assist in the preparation of IMS programs. It is possible to run COBOL code containing embedded calls to both the IMS/VS database (DB) and data communication (DC) systems. Debugging facilities are also provided. The product costs £1250 and is available now. Call Realia on 01 602 8066.

Virus Hunter

With so many people sending us things on disk, we decided it was about time to ensure that the Editor's machine was free from viruses. According to a very fast-working disk scanner called Virus Hunter, from Defiant Systems Ltd (0752 266948), it was.

Finally

Finally, a reminder that we are about to disappear for a month. There is no January issue of .EXE, so please do not ring us up to query its non-arrival. Meanwhile, we should like to thank our readers and advertisers for their support this year and, as the Editor generally puts it, wish you all a Merry One and a Happy New One.

POWER AND FLEXIBILITY

BRIEF 3.0 increases your productivity for just £149.95*

*VAT extra, FREE delivery

BRIEF has established itself as the leading programme editor on the market today. Now, with the launch of the latest enhanced version – BRIEF 3.0 – it offers even better productivity.

ENORMOUS UNDO POWER

BRIEF 3.0's enhanced UNDO capability takes the fear out of programming. BRIEF 3.0 allows you to undo any command that affects the cursor or the text (including global changes) up to 300 times, even back to the beginning of your session. It's the safety net you need to program with confidence.

REAL FLEXIBILITY

There's built-in support for 36 languages including Assembler, BASIC, C, COBOL, dBase, FORTRAN and Pascal. And thanks to its complete macro language you can make BRIEF 3.0 fit your own programming style with ease: change key assignment, commands, prompts, colours – even the indenting styles.

BRIEF 3.0 – new benefits

- With our new C-like macro language, CBRIEF, C programmers can write macros on the fly and not have to switch between C and a different macro language
- The new source level debugger works with both the original BRIEF syntax and the new CBRIEF macro language
- Maximum lines per file has been increased to 4 billion
- We've added to and improved template editing and smart indenting for ADA, Cobol, Basic, Fortran, Modula-2, Pascal and C
- We've added editable, multiple keystroke macros that can be saved and restored
- You can even save and restore your window set-ups

Plus the features that made BRIEF famous

- Flexible windowing
- Unlimited number of files
- File size limited only by desk space
- Regular expression search and replace

No other editor puts this much power and flexibility in your hands. So if you're looking for extra productivity purchase your copy of BRIEF 3.0 today.

Solution Systems

SOLUTION SYSTEMS are the UK distributors for BRIEF 3.0. When you buy from us you get the authorised UK version of BRIEF 3.0 dual media (3½" and 5¼") plus bonus macro diskette.

Special phone line for Access/Visa card holders. Save time by phoning 0763 73457 between 9.00 and 6.00 Monday to Friday to place your order.

N.B. Official purchase orders from local authorities, PLCs and educational establishments are accepted by post or by Fax.
Fax no: 0763 73460

BRIEF 3.0 The Programmer's Editor

Send to: Solution Systems UK, The Red House, 84 High Street, Buntingford, Hertfordshire SG9 9AJ

Please rush me my copy of BRIEF 3.0

☐ **I enclose cheque to the value of £172.44 (£149.95+VAT) crossed and made payable to Solution Systems UK.** **Orders despatched same day.**

NAME ____ ADDRESS ____

POSTCODE ____

☐ **Please charge my Access/Visa* account the sum of £172.44 (£149.95+VAT)**

Card No ____ **Expiry Date** ____ **Signature** ____

EXE 12/89

*Delete as appropriate

CIRCLE NO. 699

Letters

We welcome opinions on any subject that is relevant to software development, especially feedback from articles published in .EXE. Please write to The Editor at 10 Barley Mow Passage, Chiswick, London W4 4PH. Unless your letter is marked Not For Publication it will be considered for inclusion on this page.

Dear .EXE

Your article in .EXE, August '89 on MUMPS was enough to make my blood boil and stir me into writing. The entire article simply reinforces the (naive) industry view that MUMPS is no more than a joke. Even your own comments indicate as much – "warning: this code contains gratuitous GOTO s. The faint-hearted etc".

My own programming experience is primarily in C these days, although I have been raised variously on COBOL, Coral 66 and a large chunk of micro and mainframe assemblers. I use MUMPS regularly, and have also worked as a MUMPS contract programmer for a year or two, so hopefully these credentials will allow me to make an objective view on language structures.

The article is simply riddled with MUMPS practices that date back to PDP8's and 11's (great though they were) and are totally out of touch with MUMPS of the late 1980's. For a start, no MUMPS programmer should be using the ancient hieroglyphics of single letter keywords any more. Perhaps there was justification on grounds of efficiency when running on the old DEC kit, but on modern hardware with modern interpreters, full keyword is the correct way to write code. All current interpreters pre-compile and tokenise these days, so efficiency arguments are now void.

Next onto the ubiquitous GOTO. MUMPS has no need for this command, since its inherent block structuring (even in pre ANSI '84 code) makes GOTO redundant. It really does get hard to debug or edit something that contains a line like GOTO AGAIN+5 – even an assembler programmer wouldn't do that, but some ancient MUMPS programmers do. Personally, none of my own code contains GOTOs – at least I can then edit with ease of mind.

The article almost totally failed to expand on one of MUMPS' main assets– that of the built-in database handling. All arrays are held as sparse B-Tree arrays. Thus there is no requirement to declare an array size. Simply stuff a value into the required element and there it stays. Arrays may have any number of levels and may have either numeric or alphanumeric subscripts. There are functions to walk down through the tree at any level (some MUMPS also give backwards walks) or walk through at the leaf level. Since these arrays can be filed permanently on disk by simply prefixing a caret character to the array name, all traditional file handling is easy. Modern MUMPS interpreters have some of the most efficient database handling that there is to be found. It is nearly always assigned to a background task (even on PCs) and buffered to prevent unnecessary reads.

I could go on, but would need to rewrite your article from scratch (and blither on for a further three) in order to do justice to the language.

Trevor Toms
Aylesbury
Bucks

Dear Sir,

I am writing to you about Nick Hampshire's review of dbPublisher. Although Migent have been selling the product as a tool for end users, I believe its real value – as far as your readers are concerned – lies in its ability to be embedded in stand alone products.

dbPublisher was built by adding a Smalltalk front end to a very powerful typesetting system. Although the engine is hidden by the interface, its full power can still be accessed through the use of its markup language.

The product has a runtime system which is designed to allow developers to create and sell customised reports. However, because the runtime has to include the typesetting engine, developers can completely by-pass the report generation facilities and just use the engine to add full DTP quality output to all their applications.

Applications talk to the engine by creating 'Tagged' text files which are then processed by the typesetting system. In principle this is similar to driving a printer, but obviously the amount of information required by the engine is far greater. The tagging system is not as simple as Ventura's because the program does not support the idea of style sheets. However, anyone who is capable of writing a saleable application should be more than capable of creating the necessary text files.

Although I have discussed runtime prices with Migent, I am not sure whether they have been finalised. However, they were sufficiently reasonable to make including dbPublisher in our programs a viable alternative to building our own laser interface.

Andy Burnett
Marketing Director
Thought Support Systems
Pwllheli

Dear Mr Schifreen,

Let me tell you a tale of QuickBASIC 4.0 and QuickBASIC 4.5. I have a program that I use for checking the speed of code produced by various compilers (I also have versions for use with C and Fortran). I also have access to 3 PCs, all using correctly rated 8087 co-processors. These PCs run at 5, 8 and 10 MHz and have very similar configurations.

Now when my test program is compiled and run on the 5 MHz machine, both the QB 4.0 and the QB 4.5 versions run at the same speed. If both versions are run on the 8 MHz machine, the version compiled under QB 4.0 runs 17% faster than the QB 4.5 version. On the 10 MHz machine, the QB 4.5 program appears to be running in emulation mode only and, if run on another 10 MHz machine without an 8087, the two times are the same. The QB 4.0 version just speeds up in proportion to the speed of the machine on which it is run.

Thinking to force the use of the 8087, I linked both programs with the NOEM.OBJ provided. The results were just the same. However if I linked the object code produced by QB 4.5 with the NOEM.OBJ belonging to QB 4.0, the QB 4.5 version then ran at the same speed as the QB 4.0 version irrespective of the machine on which they were run.

All this sounds pretty unreasonable, but I can reproduce the results every time. It would appear that the way QB 4.5 handles

the co-processor is adrift. I've tried Microsoft UK and USA and it's like throwing things into a black hole. I did actually speak to the technical help people at Microsoft but they were no help.

Am I doing something daft? I've compiled my test program under Fortran 4.10 and C 5.1 and these run quite happily on any of the machines. The output produced by the program, no matter under what language or version it is compiled, is always the same, but the non-linear variation in program speed with processor speed is bugging me. Have I wasted my money by upgrading a compiler?

Ian Coole
Physics Department
Imperial College
London

Dear Sir,
I saw the correspondence in your October issue concerning bugs in Borland's Turbo Basic, so here is another one I have discovered.

I needed to get at the address of a string variable in order to pass it to the DOS find-matching-file function, which requires a pointer to an ASCIIZ string as its argument. The Turbo Basic manual states that all strings are stored in one segment (and we know that the string segment can become full, so this statement seems plausible). You might think that you could grab the segment address any time using VARSEG (stringvar$), and it would remain valid throughout the program's execution. Well, you'd be wrong.

After many hours of tedious effort (it's tedious because of the primitive debugging provided in Turbo Basic), it turns out that you get the right value for the segment *only* if you call VARSEG from a statement that's physically close in the source file to the position at which you need to use it. Grab it at the beginning, in amongst the global declarations and other header stuff, as I did at first, and you get garbage.

Presumably the system is changing the string segment on the fly as it goes along, or else it's using un-normalized pointers (which seems unlikely). Either way, anyone needing to do this might like to benefit from my hard-won knowledge.

Andrew Duffin
Computer Co-ordinator
Beecham Pharmaceuticals
Irvine
Scotland.

Dear .EXE,
Readers who use dBASE frequently may be interested in the SET ODOMETER command, which is not documented in any of my manuals. This command allows you to specify how frequently the screen is updated during a COPY command. Normally, the record counter on the screen is incremented after each record is copied. SET ODOMETER TO 50, however, will cause the counter to count in steps of 50, which can result in quite a large time saving. When sorting a large file, setting the odometer to 20 can halve the time taken to perform the sort.

Colin Chapman
Bury
Manchester

Robert Schifreen replies:
I've tried this command on a copy of dBASE III Plus and it does indeed work as you describe. The dBASE IV manual documents the command, saying that permissible arguments are integers between one and 200 (or you can turn off the updating altogether with SET TALK OFF). Some brief experimenting with the command leads me to suspect that a value of around 25 or 30 is the optimum setting. Before you use SET ODOMETER in your programs, though, remember that there may be a good reason why Ashton Tate did not document the command until version IV – there may be obscure bugs that only show up in certain circumstances.

.EXE

The Man who is C++

The C++ language is an idea whose time has come. Bjarne Stroustrup is the man who had the idea, Paul Smith is the man who talked to him.

Providing an introduction to Bjarne Stroustrup may seem superfluous, but one must always cater for the possibility that Rip van Winkle has just taken out a subscription to .EXE Magazine. Along with such figures as John Backus, Denis Ritchie and Niklaus Wirth, Stroustrup is a major influence in the design of modern computer languages. In particular, his C++ language, an object-oriented superset of C, looks set to dominate programming in the 1990s in the same way that C has reigned in the late 1980s. His book, *The C++ Programming Language*, showed up its illustrious predecessor 'K&R' in a new light: as a light read. Despite his thoroughness and attention to detail, making few concessions to those less quick than himself, Stroustrup's book is not without humour; as when he directs his readers to George Orwell's Newspeak, as described in *1984*, for an interpretation of the name of his language.

Still working at the same AT&T Bell Laboratories which fostered both the C language and the UNIX operating system, Stroustrup has recently completed the first major revision of his C++ specification. As the first Version 2.0 C++ compilers begin to appear from manufacturers, it seemed like the right time to call up the Master and get his thoughts on the state of play.

Just how did you come to get into language design and C++ in the first place? What is your background?

I got my Masters degree at the University of Aarhus in Denmark. Aarhus is my home town. I specialised in Operating Systems and machine architecture, doing things like instruction set design and microprogramming (the programming of a machine's fundamental instructions at the register transfer level). I got my PhD from Cambridge [University, England], where I worked on architectural support for distributed systems. My major tool for experimentation was a distributed systems simulator that I wrote. PhD students did not have seven-processor systems to play with in those days, and anyway, controlled experiments are **very** hard to do on real systems with people doing real work. The simulator was initially written in Simula67. This language is one of the most beautiful and most innovative ever designed. From this stems most of the key concepts of what has become known as 'Object-oriented programming.' Simula67 provided classes and class hierarchies (also known as 'inheritance'), virtual functions and strong type checking. I think that it was Tony Hoare who commented that 'Simula was not only a major improvement on all of its predecessors but also on most of its successors'. I'll second that!

There was a major problem, though. Having gone through the unusually pleasant experience of designing and debugging this simulator in Simula67, I could not afford to run it. Simula67 provides the essential mechanisms for expressing a design (classes) without which I couldn't even have conceived of the final program and the strong (static) type checking without which I could never have gotten the bugs out that easily. However, it was very demanding of run-time resources: both time and space.

I had very few choices: I could leave Cambridge (without a PhD), I could wait 6 years for hardware to get cheap and fast enough for me to afford it, I could get a very rich sponsor, or I could rewrite the simulator into a language that would run fast on a small machine. Only the last alternative seemed feasible, so I rewrote the simulator in BCPL and ran it on the experimental CAP computer.

It was a most unpleasant experience; BCPL makes C look like a very high level language. However, the basic fact was that after a lot of work, I got my data. I swore that I would never again approach a project of this complexity without proper tools.

My idea of a proper tool for such projects had gone through a significant transformation as a result of this experience. Such a tool (a language plus its related compilers and so on) must help cope with complexity and help ensure correctness of the programs written. In other words, I insist on strong type checking and classes. It must also be affordable in terms of hardware resources (both run-time and space), it must be 'open' in the sense that it must be easy (and cheap) to use other peoples' software from it (such as FORTRAN libraries, C graphics systems, databases). Also, since I work on many kinds of computers, the implementation of this tool must be very portable, and it must be possible for me to write very portable programs in it.

Unfortunately, such a language did not exist. That left me in an awkward position when I later (1979) needed to start another demanding project at Bell Labs.

C++ is born

I then decided to build such a tool; this became first 'C with Classes' and then C++. During this design and implementation phase a couple more 'rules of the game' were formulated. I wanted a language that could be efficiently implemented on traditional hardware by someone with limited time for the job (me) and without the need for really clever implementation tricks (which I might never get right); this could also be seen as a corollary to the portability requirement.

I decided on C++ being 'as close as possible to C – but not closer' to ensure that I did not risk designing unexpected limitations into the language, so that I could apply C tools (such as linkers, profilers, editors and so on) to C++ programs, and to decrease significantly the burden of teaching the 'new stuff'. I also observed that C was basically OK when it came to expressing computation; its deficiencies are in the area of program organisation and support for good design and programming – those then became my areas of work.

It might be worth mentioning that C is often unfairly maligned for not solving problems it was never designed to solve. C was designed to raise the level of language used in systems programming, to enhance portability significantly, and to eradicate assembly language programming in most branches of systems programming. In this, it succeeded beyond all expectations. Many of the things we take for granted these days are direct results of C's success in these areas.

If you call Cfront a preprocessor, you should also call an assembly code-generating C compiler a preprocessor

I had no wish to 'improve' C in such ways that these achievements were endangered. To my mind there is little advantage of using a language that is so 'high level and clean' that it forces people to use assembler on a regular basis. Consequently, C++ is designed to span a larger spectrum of programming tasks than most languages.

How long was the development of C++ 2.0, and how many people were involved?

Essentially, 2.0 was grown over the years after release 1.0 in 1985. We had multiple inheritance, the feature people most directly associate with 2.0, working in May of 1987, but the final push that produced 2.0 was done from late 1988 until June of '89. There were about a hundred people involved in the reference manual review, dozens of people took part in activities such as bug-report screening, documentation, arguing with me over features, trying out early versions and so on. The core group consisted of Stan Lippman and me, who did almost all of the coding, Pat Philips, who handled integration, Andy Koenig who did testing, and Barbara Moo, who coordinated the various efforts.

In your book The C++ Programming Language *(Addison-Wesley 1986), you say 'C++ was designed to be used in a rather traditional compilation and run-time environment, the C programming environment on the UNIX system. Features such as exception handling or concurrent programming that require non-trivial loader and run-time support are not included in C++. Consequently, a C++ implementation can be very easily ported.' Do you now regret these design decisions?*

I strongly believe that those were correct decisions. Some people seem to think that a pragmatic view precludes principles. That is not the case. I know my principles of programming languages and of programming about as well as anyone else, but I am very reluctant to let theory dominate my thinking. Far too often we have seen notions of religious purity blinding people to hard facts of life and imposing preposterous constraints on people. A pragmatic view and an understanding of the problems of the users are essential – as is the need to consider every language feature in a larger context so that a language doesn't simply become a random collection of 'neat features' but has an internal consistency and logic. The 'shopping list approach to design' simply doesn't work and C++ was certainly not designed that way.

My ideal for the design of C++ is an organic growth from a solid base guided by both principles and experience. I think that is what has happened up till now and I see that continuing for a while longer.

Naturally, some aspects of C are a pain in the neck. Departing from them would have made the design of C++ cleaner and its implementation and use easier. For example, the C declarator syntax is an experiment that failed and some aspects of the C array concept are beyond repair. However, I consider these issues second order compared with such matters as efficiency, portability, compatibility and the ability to coexist with an unbelievable diverse set of tools, systems, languages, design approaches and so on.

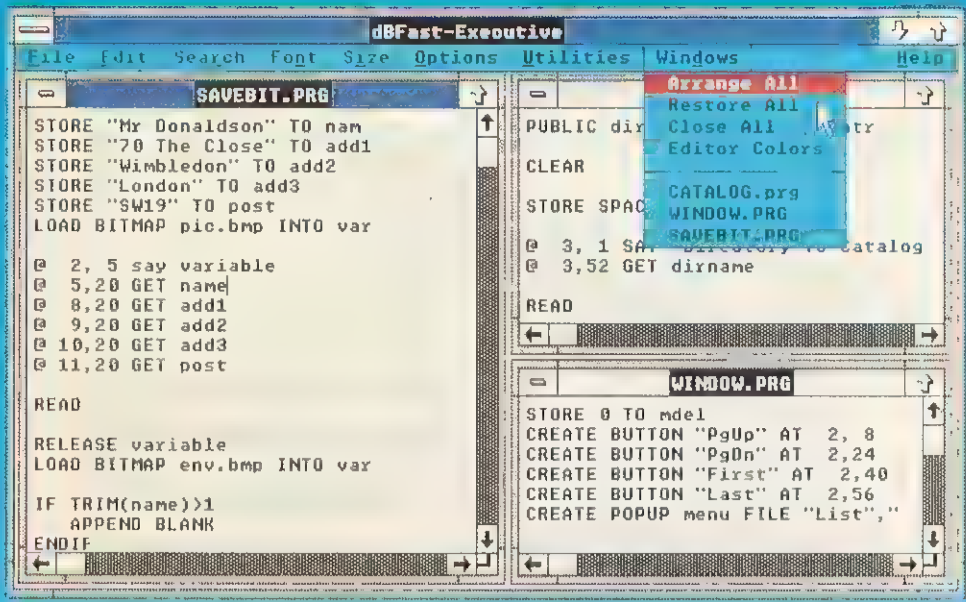
Is C++ going to become standardised, with a committee sitting to determine the language definition and C++'s features? Would you be happy to see C++ 'move upstairs' in this fashion?

The ANSI standardisation of C++ starts in December. A formal standardisation process can bring a lot of good in terms of clarity of definition, agreement on details, stability for use, and providing a stable

NOW! Develop dBASE applications under Microsoft Windows

dBFAST/Windows

Now there is a complete dBASE language development environment running under MS-Windows. Now it is possible to add the user-friendly Windows interface to dBASE and most Clipper applications. Language extensions allow implementation of the Windows interface even during development. Design multiple windows, pull-down menus, check-boxes, buttons, bit map pictures and other graphics for more "user friendly" applications. dBFast/Windows contains an interactive editor, compiler, runtime system and other utilities. You don't need to buy dBASE!



Full Featured dBASE programming environment

dBFast/Windows combines the most powerful tools available for developers, the dBASE language and Microsoft Windows. Using familiar dBASE syntax, you can:

- ✓ Develop dBASE applications under MS-Windows
- ✓ Create, compile and debug in a single screen
- ✓ Use macros, arrays and UDFs
- ✓ Support Dynamic Data Exchange (DDE)
- ✓ Use existing dBASE and Clipper applications
- ✓ Distribute applications *without runtime fees*
- ✓ Port dBFast/Windows applications to Apple Macintosh
- ✓ Execute up to 15 times faster than interpreted dBASE

30-DAY MONEY BACK GUARANTEE

Our philosophy is simple. There is only one way of evaluating a product and that's to try it. So, like all other database products that MEGATECH distributes, dBFast/Windows as well as dBFast/MAC carry a 30-day money back guarantee. The package contains a "locked" but fully operational version of the program as well as a (sealed) "unlocked" version. As long as the seal is not broken and the package is returned to us in good condition within the 30-days we will refund your purchase price. Immediately. No questions asked. So try it. Send for more information or phone or fax us your order today!

Use familiar dBASE syntax under Windows!

It is easy to create a new application under dBFast/Windows. You'll be using dBASE commands and functions so it will feel intuitive. The Windows interface is a natural, too. Menus and buttons for example can be created with a single line of code. (See screen WINDEMO.PRG) In most cases, the tools for creating the Windows interface are available through new commands and functions rather than changing the way existing dBASE commands and functions operate. The documentation carefully outlines the graphic options. You can also port existing dBASE and Clipper applications to dBFast/Windows. (Some minor modifications may be needed before adding the graphical interface.)

Then port your applications to the Macintosh!

dBFast/Windows offers complete portability (including data and index files) to the Macintosh. Once an application is complete under dBFast/Windows, it can be ported to the Macintosh with dBFast/MAC. Language syntax and file compatibility allow you to move to the Macintosh by just re-compiling programs and transferring data files. Of course, the same holds true if you develop on the Macintosh using dBFast/MAC. Then, you can port your Mac applications to the IBM platform via dBFast/Windows. In either case, you save time and money by sharing applications over two platforms!

CIRCLE NO. 701

To: MEGATECH, 111-113 Wandsworth High Street, London SW18 4HY

Please send more information:

Please supply:

___ Copies of dBFast/Windows @ £249: £______

___ Copies of dBFast/Mac @ £249: £______

Handling & 1st Class Shipping £ 6.00

Add 15% VAT £______

I enclose a total of: £______

Payment:

Cheque ☐ Access ☐ Visa ☐

Card No: ______

Card Expiry Date: ______

Card Name ______

Card Address ______

______ Postcode ______

Daytime Tel No: ______

Signature ______

A VAT invoice will be included in the Parcel. Enclose a company letterhead if invoice name and address needs to be different from card name and address.

MEGATECH

111-113 Wandsworth High Street, London SW18 4HY.
Tel: 01-874 6511, Fax: 01-877 1384

dBFast Products are also distributed by GEM Software Distribution (0279-442842) and NEWSTAR Software (0245-265017)

base for further evolution. I hope that is what is going to happen and I think we can make that happen. On the other hand, some standards committees have turned into language design committees and become vehicles for rampant featurism, 'design' by vote of anyone who cares to turn up and horse trading ('if you vote for my feature, I'll vote for yours'). Such a horror show must be avoided.

Do you regard version 2.0 of the language as the beginning of a move away from its roots?

Not at all. If you look at my book you will see multiple inheritance, parameterised types and exception handling mentioned as areas where we needed improvements, but where I did not at the time know of solutions that met my rather stringent criteria for inclusion into C++. Only when we have all of those features will C++ have expanded to fill its original scope.

C++ versus ANSI C: has the development of the C language split into two separate threads, or will C++ and ANSI C merge again in a future 'super C'?

I think C++ already is such a 'super C.' Many ANSI C features have been adopted – as many as could be safely absorbed without damaging the type system. It is also worth noting that the ANSI C committee's work on the standard C libraries can be directly adopted by C++.

Conversely, ANSI C adopted variants of C++'s function argument type checking and type conversion, the C++ function definition syntax, and a variant of C++'s `const`. C++ is already closer to ANSI C than most people think.

How would you characterise the significance of version 2.0 of C++, from the point of view of the original designer of the language? What in version 2.0 is most important?

Release 2.0 of C++ was made generally available by AT&T on June 30, 1989. Release 2.0 is the first major revision of the C++ language and will provide a base for formal standardisation and further evolution of C++.

The most important thing about release 2.0 is not the new features it provides, but the improved quality of the compiler, the libraries, the documentation and, in particular, the C++ reference manual.

The manual is the result of much work within AT&T and of a review by about a hundred people from very diverse organisations world-wide. Experienced C++ users from academia, industry, and government and C++ compiler developers from many organisations (that usually compete vigorously) all helped.

You have never seen an AT&T advert for C++ – its current popularity was achieved without expensive commercial hype

After a further minor revision and clean-up, I plan to publish this manual together with additional information about implementation techniques and reasons for design decisions as a book. It should be finished near the end of this year.

The definition of C++ 2.0 now provides a common base for all C++ compiler and tools writers and will become the starting point for ANSI standardisation of C++. Release 2.0 removes all non-essential incompatibilities with ANSI C leaving C++ for most practical purposes a superset of ANSI C.

Most new features in 2.0 take the form of removal of restrictions. Together they allow for much cleaner code, a more natural programming style and eliminate many common programmer errors.

It might be worthwhile pointing out that I still control the definition of C++. I designed every one of the 2.0 features, wrote the manual sections for them, and did the initial implementation of them.

Taking exception

Do you still believe that exception handling belongs outside the language?

I have pointed out before, in various papers, that we need a better mechanism for handling errors. However, we cannot accept a mechanism that would damage run-time efficiency, portability, or the ability to use C++ in mixed language programming.

So, though we have acknowledged the need for years, we have had no acceptable solution, let alone a proper design. This is changing. On October 6, Andrew Koenig and I presented a paper 'Exception Handling for C++' to the C++ at Work conference in Boston.

The scheme is based on the notion of a 'try block', in the style of the Modula-2+ and Modula-3 syntax, with the distinct C++ touch of having exceptions be objects of a class `exception`. This enables us to bring all the power of the language to bear on exceptions in areas such as type-checking, lexical scope, and extensibility of the `exception` concept. (Since `exception` is a class, users can derive their own variants of the class from it. See Figure 1.)

It is important to realise that this exception handling mechanism is designed as a proper language facility so that it allows several alternative implementations. In particular, we can implement it using `setjmp()`/`longjmp()`, thus getting the portability we need. Initial measurements show that this style of implementation is likely to be unacceptably slow in some contexts. To alleviate this, we also have a second technique that yields zero run-time overhead when an exception is not raised. This technique cannot be implemented in a portable way, though. We imagine that people will use the portable, but relatively slow,

```
    int f()
    {
        try {
           return g();
        }
        catch (xxii) {
           // we get here only if 'xxii' was raised
           error("g() goofed: xxii");
           return 22;
        }
    }

Somewhere in g() or some function called by g() we can then raise
the exception like this:

        xxii.raise();

        raise() is an operation on objects of class exception.
```

Figure 1 – Exception handling using the 'try block' scheme

implementation until a variant of the efficient implementation becomes available on their particular system.

This exception handling mechanism is not available from anybody and I expect it will remain experimental and under discussion for some time.

AT&T's implementation of C++ comprises of a Cfront preprocessor to a standard C compiler. Is it satisfactory to require an extra preprocessing step before compilation?

Describing Cfront as a 'preprocessor' is completely misleading. It never was a preprocessor in the same sense as the C preprocessor or RATFOR. Cfront is a classical compiler that does a complete syntax and semantic analysis before producing output. It produces C simply because C is a very widely available interface to good code generators. If you call Cfront a preprocessor, you should for consistency also call an assembly code-generating C compiler a preprocessor.

One would expect code generated from Cfront to be less efficient than code generated by a compiler without the intermediate C step. Until now, this has – surprisingly even to me – not been the case. The speed and space records on all machine architectures are still held by some Cfront/C-compiler combination. I expect and hope that the 'native' C++ compilers will soon progress to the stage where they beat Cfront in generated code quality, but don't expect spectacular improvements over Cfront because the Cfront/C-compiler combinations already out-perform many C compilers for the C subset of C++.

Doesn't this dependence on a C compilation step limit the design of C++?

The C++ language does not depend on a C compilation step, the Cfront implementation of the C++ language does, and no, the Cfront two-pass compiler technology has not constrained the design of C++. I have always known the distinction between a language and its implementations. The initial implementation of C++ (Cfront) has not been allowed to constrain the design of the C++ language. The constraints on the C++ design has been C compatibility and the generally poor state of linker technology – and then of course the first order design criteria of type-safety, low-level efficiency, and ability to fit into traditional run-time environments.

People sometimes point to debugging as an area that shows that C++ is deficient because of the Cfront implementation technique. The appearance of C++ level source language debuggers for C++ implemented using Cfront from Apollo, HCR, Sun, Apple and ParcPlace should put this issue to rest. The C++ debuggers used inside AT&T have not been for sale.

Figure 2 – Generic classes implemented as 'templates'

```
template<class T> class Vector {
        T* p;
        int sz;
public:
        static exception range;
        static exception bad_size;

        Vector(int i)   // constructor
        {
                if (i<0 || sz<HUGE) bad_size.raise();
                p = new T[sz=i];
        }
        ~Vector() { delete p; } // destructor

        T& operator[](int i)    // subscript operation
        {
                if (i<0 || sz<i) range.raise();
                return p[i];
        }

        int size() { return sz; }
};

You can now declare Vectors of any type you care for like this:

        Vector<int> vi(20);
        Vector<complex> vc(100);
        Vector<shape*> vs(20);
```

Actually, C compatibility in general and the Cfront technique of generating C as our assembly code has been very important. C compatibility allowed me to have a complete language, a very large set of libraries (the C libraries), and high quality code generators immediately. Also, because C was widely known, the burden of education was greatly diminished. In all, this allowed me to support users within six months of starting enhancing C. It seems that I started a trend. People are now providing C generating compilers for languages such as Modula-3, Ada, and Common Lisp. C is becoming the universal intermediate language for people interested in portability.

Is it then the case that C++ is, from an implementation point of view, quite independent of C; that you have made it compatible by choice, and that there is nothing in C++ that has to be there simply because of your use of C as intermediate code between the C++ compiler and the final object code?

Exactly! 100% correct. If you need an added argument for that position: there exist at least four available and five almost complete C++ compilers that do not use C as an intermediate form.

Do you agree that C++ should have had proper generic classes, not a `#define` bodge? If so, why didn't you supply them?

I certainly agree that generic classes are desirable. However, you must remember that C++ is one of the languages that had to work hard for a living during its childhood. In this, it resembles FORTRAN and C (both of which I have the greatest respect for) and differs from languages such as Ada, Algol68, and PL/1, that came into existence as complete – if unimplemented – designs. My book was written and release 1.0 of Cfront was done not because C++ was perfect, but because C++ had reached a plateau of its development where it was reasonable to allow more people to use it. There was quite a clamour for it at the time (as there has been ever since) and there seemed no reason to keep C++ out of use simply because it could be even better.

Throughout its early life, C++ was a very low budget operation that had to pay off immediately in terms of benefits to real users. I know that this does not conform to the view some people have of AT&T as a huge organisation capable of throwing endless time, money and manpower at problems, but it does actually fit well with the way work is done in the computer science research centre at Bell Labs. Note that you have never seen an AT&T advert for C++. The current popularity of C++ was achieved as a grass-roots movement without expensive commercial hype.

Also, and most importantly, I believe that languages ought to grow guided by both general principles and experience. Without users there cannot be relevant experience and without principles you get nothing but a mess. So C++ evolved through stages: First C with Classes in 1980, then Release E in 1984, and release 1.0 in 1985. The current release 2.0 builds directly on

these. The 'generic classes' you ask for will most likely become the next major step together with exception handling. (See example in Figure 2.)

Another reason for leaving templates out of release 1.0 was that at the time we were not sure that we understood the compilation process well enough to build something that both ran fast enough and produced sufficiently good code.

It is worth noting that among the so called object-oriented languages, C++ is unique in taking efficiency seriously to the point on meeting C and FORTRAN head on. Most other languages rely on quality, productivity improvements, and/or glitz to overcome a burden of inefficiency compared to C; C++ has always had to be equal to or better than C. This affects C++ strongly. For example, C++ has genuine static and automatic (ie on stack) variables where other 'object oriented' languages rely on pointers to objects to free store. This saddles these languages with an overhead of memory management and pointer chasing that is avoided in C++. The flip side of this is a bit more work for the C++ programmer and quite a lot more work for the C++ compiler writer. The result is pleasing, though. The difference in run-time efficiency is usually measured in factors – not fractions!

C++ and UNIX

A recent rumour suggested that C++ was being used to write parts of the next release of AT&T UNIX. Is C++ displacing C as the UNIX systems programming language already? If not, how soon do you expect it to do so?

Well, the rumour was definitely true, but I don't know if it still is. There was, and maybe there still is, a joint AT&T/Sun project to evolve UNIX System V. It was/is being done in C++. Whatever is going on with that particular project – which is of course mostly dependent on grand corporate politics – there are definitely active UNIX kernel projects that use C++.

Yes, C++ is already displacing C as the UNIX systems programming language in some places. However, it would be plain silly to expect C to go away any day soon. After all, one of the principles of reuse is 'if is ain't broke, don't fix it'. Since UNIX by and large is not broken, we shouldn't start messing around with it simply to convert it to C++. Only if we want to replace or perform surgery on some subsystem would we start using C++.

If you are talking of System V.4, which I believe is soon to be released, then the answer is that no kernel work was done in C++; but I believe that some utilities (sdb springs to mind) are C++ programs.

Once you have C++ 2.0 out of the way, and you have finished your new book on the subject, what do you plan to do next?

Have a long vacation! Seriously, though, I have to write two books (an annotated C++ Reference Manual and a revision of my book) and there is still exception handling and parameterised types to do. That should keep me busy for quite some time. The ANSI standardisation of C++ will also demand work and then there are users to help. I think it is a bit early to speculate about 'what next'. EXE

Many thanks to Bjarne Stroustrup for taking time to give this interview. Paul G Smith is a specialist in graphics, communications and the application of object-oriented programming techniques. He can be contacted on CIX as 'pgsmith', and on AppleLink as 'UK0310'.

We do not have sufficient space here for a bibliography of relevant papers by Bjarne Stroustrup. You will find one posted on CIX in the conference 'exe/files', or, alternatively, please send an SAE to the editorial address.

The Thinking Programmer's Guide to UARTs

Did you know that by replacing, at the cost of a tenner, one socketed chip in a standard PC/XT, you can make it run software that would tax a 16 MHz 386? Andrew Margolis knows: fortunately he is telling.

There are three different types of UARTs (Universal Asynchronous Receiver/Transmitter – the chip that drives the serial port) available for PC-compatible machines. All of them are manufactured by National Semiconductor. PC/XT clones generally use the 8250A, AT class systems come with a 16450 and newer PS/2 MCA-compatible machines are supplied with the 16550. If you buy add-on serial cards for a computer, the ones advertised as 'PC/XT compatible' will have the 8250A on board, those advertised as 'AT compatible' will have a 16450. Cards with the 16550 are more difficult to find for the PC bus, but appear to be the standard for MCA systems.

The 8250A was designed in the days when the power user had a 4 MHz Z80 machine on their desk. Consequently, it is not very fast. For instance, the documentation shows that it takes 1000 ns to reset the interrupt line after any interrupt service, and a similar length of time to react after being told to change the state of one of the modem output lines (such as DTR). Similarly, the minimum read and write cycle times are 755 ns: this is a long time in processor terms.

CPU MHz	Nanoseconds per clock
4.77	210
6	167
8	125
10	100
16	63
20	50
32	31

Figure 1 – CPU speed, expressed as nanoseconds per clock period

Figure 1 shows the length of one clock period, in nanoseconds, for a range of CPU speeds. Bearing in mind that a single I/O instruction on an 8086 takes at least 10 clocks, a 4.77 MHz PC will take a minimum of 2100 ns for an access to a UART register. Clearly, this isn't stretching the performance of the 8250A. On the other hand, a 10 MHz system will only take 1000 ns, which is approaching the point at which the performance of the UART is going to slow down the system, in the same way that slow DRAMs on memory boards can degrade the performance of fast 386 systems.

Unfortunately, it does not follow that a fast CPU will guarantee modem compatibility

The 16450 is simply an improved specification version of the 8250A, with better timings to take advantage of a faster bus. It is so compatible that software cannot tell which type of UART is being used on a given system. The timings given earlier for the 8250A are much improved for the 16450. The 1000 ns period to reset the interrupt line is cut to 200 ns, while the read and write cycles are halved to 360 ns:other timings show similar improvements. Having said that, it ought to be remembered that while the speed benefits of faster RAM are cumulative and become noticeable fairly quickly, there is often no benefit at all in being able to pick up characters a few hundred nanoseconds quicker when you are running at normal baud rates. You simply spend longer waiting for the next character; picking up the current character at twice the speed does not make the next one arrive any earlier. If you are downloading large amounts of data, the only saving you will make is a few nanoseconds picking up the very last character.

The only situation where faster speeds are cumulative is where the UART is being run as a background task. For instance, MS-DOS's background PRINT utility, used with a serial printer, should, in theory, work faster when a 16450 is substituted for an 8250A. In practice, I have been unable to detect any difference – even when printing large quantities of data.

Generally, a well designed serial board will be able to handle any of the four possible permutations of AT or PC bus and 8250A or 16450 UART equally well. The two chips' timings are, in fact, given as maximum and minimum values. It's quite possible that an 8250A, performing at its peak, might be equivalent to a 16450 which is having a bit of bad luck with bus cycles. There are tales of 16 MHz systems failing to work with V21/V23 modem cards designed for 4.77 MHz PCs. In these cases, it is most unlikely that replacing an 8250A with a 16450 will solve the problem. The design of the modem card and its firmware are much more likely to be at fault.

Blame the CPU

High speed communication problems are not solved by replacing the 8250A with the faster 16450. This is because it is not the UART at fault – it is the CPU. Figure 2 shows a fragment of assembler code for an interrupt service routine for a UART. It collects characters as they come in and places them

Figure 2 – Partial 8250A Receive ISR

```
; fragment of receive character ISR
;
; data areas
;
tempest db 2 dup (?)
; storage for stack segment
;
tempist db 2 dup (?)
; storage for stack pointer
;
        db 32 dup (?)
intstak equ $ ; base of stack
;
inbuf db 2048 dup (?)
; base of circular buffer
;
nextin db 2 dup (?)
; offset base to next store
;
;     memory accesses —> ————————┐
;     clock periods ——> ——————┐  │
;
isr:                       ; 62  7
;
; save stack registers and load ours
;
   mov   cs:tempest,ss     ; 20  1
   mov   cs:tempist,sp     ; 20  1
   mov   sp,cs             ;  2
   mov   ss,sp             ;  2
   mov   sp,offset intstak ; 10
;
; save register we need
;
   push  ds                ; 10
   push  si                ; 10
   push  ax                ; 10
   push  bx                ; 10
   push  dx                ; 10
;
   push  ds                ; 10
   pop   cs                ; 10
;
; see whether a char is available
;
   mov   dx,tps            ; 10  1
   in    al,dx             ; 10  1
   and   al,00000001b      ;  4
   jz    exserv            ;  4
;
; if character, read it, else exit
;
   sub   dx,5              ;  5
   in    al,dx             ; 10  1
;
; store in circular buffer
;
   mov   bx,offset inbuf   ;  4
   mov   si,nextin         ; 14  1
   mov   [bx+si],al        ; 17  1
   inc   si                ;  3
   and   si,2047           ;  4
   mov   nextin,si         ; 15  1
;
; end of interrupt, restore and exit
;
exserv:                    ;
   mov   al,20h            ; 10
   out   20h,al            ; 10  1
;
   pop   dx                ; 10
   pop   bx                ; 10
   pop   ax                ; 10
   pop   si                ; 10
   pop   ds                ; 10
   mov   ss,cs:tempest     ; 21  1
   mov   sp,cs:tempist     ; 21  1
   iret                    ; 32  3
;
;  instructions           = 34
;  clocks                 =    430
;  memory accesses        =        21
```

in a circular buffer. The listing has been annotated to include the number of clock periods that each instruction takes and the number of memory accesses. The routine has some frills (maintaining its own stack, keeping the base address of the UART as a memory variable) which could be removed to speed it up. On the other hand, it does not include some items which might be essential: there is no flow control, no checking for buffer overflow and it does not cater for any interrupts except received data. However, it is a reasonable sample of a minimal program of its type.

The instructions take 430 clock periods to execute. There are eighteen 16-bit memory transfers, three I/O instructions and one instruction fetch for each of 34 instructions: the amount of time these take varies with the processor and with the number of wait states. For instance, an 8088 takes four extra clocks for each 16-bit memory transfer. Conservatively, we can assume that each character coming in and generating an interrupt will take up about 550 clock periods.

The second column of Figure 3 shows how many clock periods are nominally available between incoming characters at a selection of baud rates for a 4.77 MHz PC (figures calculated for 10-bit words made up of 1 start bit, 8 data bits and one stop bit). However, we can't assume that the CPU is being driven at maximum efficiency. According to *The 8086 Book* (Rector and Alexy, Osborne/McGraw-Hill, ISBN 0-931988-29-2), 'a single 8086 CPU can utilise between 50% and 80% of the available bus band width.' Here is another consideration: we don't want to lose characters when a higher priority interrupt is generated. Remember that a clock tick, a key press or *both* could occur at any time. Let us assume that the timer and keyboard interrupt take about the same time as a receive character interrupt. We will also allocate an equivalent amount of time for the application program that is running (after all, that is what the computer is supposed to be doing). We end up with only one-eighth of the nominal clock periods per character received for our interrupt service routine. It is this more practical figure that appears in column 3 of Figure 3. Half the available clock periods have vanished, due to bus overheads; the rest are shared between two other possible interrupts and at least one application.

You will recall that our minimal interrupt service routine takes 550 clock periods to run. Figure 3 shows that the standard IBM PC can't be expected to handle serial ports reliably at speeds faster than 9600 baud. (The term 'baud rate' is generally used – as it is throughout this article – to mean 'bits per second'. This is not strictly correct. Baud rate officially refers to the frequency of electrical impulses on the communications line. Phase shifting allows more information to be encoded in each impulse; thus a 2400 bps modem may be running at 600 baud. However, this is not an issue when considering communications with the PC serial port.) The table extends the calculations involved for other CPU speeds, indicating that you can't guarantee being able to run at 19200 baud on a machine slower than 10 MHz.

It is important to emphasise that these calculations cannot be absolute. Hardware design affects the efficiency of the CPU, and a decently coded BIOS running in shadow RAM will minimise the time taken by timer and keyboard interrupts. For example, if the hardware gave us an 80% availability of clock cycles, and the keyboard and timer interrupt service routines together took up no more time than the incoming character routine, then we could double the maximum baud rate feasible on any particular system. Conversely, if we added routines for flow control and enabled other UART interrupts apart from receive data, then our routine would take up more time and the maximum baud rate would be lower. On the same machine, one package might run easily at 19200, while other software might struggle at 9600. Figure 3 gives a reasonable idea of relative power and shows how much time communications can take up, but it can't be said to be accurate. Use it as a rough guide to what might be feasible on a particular system.

Speed in bits per second	Clocks per word on a 4.77MHz PC	Assume 50% bus efficiency and 25% CPU usage	——> same for different CPU speeds					
			6MHz	8MHz	10MHz	16MHz	20MHz	32MHz
300	159000	19875	25000	33333	41667	66667	83333	133333
600	79500	9938	12500	16667	20833	33333	41667	66667
1200	39750	4969	6250	8333	10417	16667	20833	33333
2400	19875	2484	3125	4167	5208	8333	10417	16667
4800	9938	1242	1563	2083	2604	4167	5208	8333
9600	4969	621	781	1042	1302	2083	2604	4167
19200	2484	311	391	521	651	1042	1302	2083
38400	1242	155	195	260	326	521	651	1042
115200	414	52	65	87	109	174	217	347

Figure 3 – Maximum clock cycles available for Interrupt Servicing

CALL FOR FREE 60 PAGE CATALOGUE

ASSEMBLERS
386 ASM Link & Locate	£425.00
MASM v.5.1	£89.00
OPTASM	£75.00
Turbo Assembler/Debugger	£89.00

BASIC
BASIC Compiler	£180.00
BASIC Interpreter	£215.00
Quick Basic	£60.00
True Basic	£60.00
Turbo Basic	£60.00

C
Lattice C Compiler v.6.0	£160.00
Microsoft C Compiler v.5.1	£250.00
Watcom C Optimizing	£199.00
C-terp	£180.00
C86 Plus	£299.00
Eco C88 with dev. library	£49.00
Express C	£49.00
High C-286	£360.00
High C-386	£550.00
Instant C	£299.00
Instant C/16M	£480.00
NDP C	£360.00
Quick C	£60.00
Turbo C v.1.5	£49.00
Turbo C v.2.0	£79.00
Turbo C Professional	£159.00
UserSoft/C	£150.00
Zortech C Compiler	£50.00

C++
Advantage C++	£310.00
Guidelines C++ with book	£185.00
Intek C++ 80386	£310.00
Zortech C++ Compiler	£90.00
Zortech C++ with source	£160.00
Zortech C++ Compiler	£90.00
Zortech C++ Debugger	£90.00

COBOL
Microsoft COBOL	£565.00
Real Menu	£210.00
RealCICS	£750.00
Realia COBOL	£750.00
RM/Cobol	£575.00
RM/Cobol-85	£750.00
RM/Screen	£240.00

dBASE
Clipper Summer '87 (U.K.)	£325.00
DB Fast/DOS	£65.00
DB Fast/PLUS	£155.00
DB Fast/WINDOWS	£155.00
dBASE III Plus	£395.00
dBASE IV	£395.00
dBMAN V	£120.00
dBXL Diamond	£120.00
dBXL/Lan	£375.00
Foxbase+ v.2.10	£245.00
Foxbase+/386	£375.00
Foxbase+/LAN	£375.00
Foxbase+ Runtime	£435.00
Foxbase+ SCO XENIX	£495.00
Quicksilver Diamond	£375.00

DATABASE MANAGEMENT
Advanced Revelation	£575.00
AskSam	£175.00
Clarion Personal Developer	£99.00
Clarion Professional	£425.00
Magic PC	£120.00
Omnis Quartz	£480.00
Opus One	£240.00
Oracle for 1-2-3	£120.00
Paradox 2.0	£439.00
PFS Professional File	£149.00
R:BASE Ext. Report Writer	£179.00
R:BASE for OS/2	£545.00
R:BASE for DOS	£439.00
Reflex	£89.00

FORTRAN
Microsoft Fortran v.5.0	£285.00
Fortran-F77L	£289.00
NDP FORTRAN-386	£360.00
ProFortran	£235.00
RM/Fortran	£360.00
SVS Fortran	£420.00

OTHER LANGUAGES
ACTOR	£310.00
CCSM Mumps Single User	£65.00
Neural Network Explorer	£120.00
PC Logo	£90.00
QNial	£90.00
RPG II Complete	£995.00
Smalltalk/V 286	£120.00
Snobol 4+	£59.00
Trilogy	£60.00

PASCAL
Microsoft Pascal	£180.00
Oregon Pascal-2	£149.00
ProPascal	£235.00
Metaware Professional Pascal	£545.00
Metaware Prof. Pascal 286	£360.00
Turbo Pascal v.5.5 with objects	£89.00
Turbo Pascal Professional	£159.00

PROLOG
Arity Combination Pack	£665.00
Expert System Dev. pack	£179.00
Intelligence Compiler	£299.00
MicroPROLOG Professional	£425.00
MProlog	£299.00
Professional Prolog II Plus	£899.00
Professional Prolog	£545.00
Prolog Advanced Toolkit	£90.00
Prolog Compiler	£120.00
Prolog Compiler/Interpreter	£399.00
Prolog-86 Plus	£150.00
SQL Development Pack	£179.00
Turbo Prolog	£75.00

AI AND OBJECT ORIENTED
C-Talk	£90.00
C+ Objects Vol 1 with Source	£275.00
CLOUT	£150.00
CxPERT	£240.00

COMMUNICATIONS
AdComm	£179.00
C ASYNCH MANAGER	£99.00
Communications Library	£139.00
Comm Pak with Breakout	£185.00
Multi Comm	£89.00
Net Lib	£120.00
NET-TOOLS	£90.00
SilverComm	£90.00
SilverComm Prog Plus	£90.00
SilverComm Combo	£90.00
Turbo ASYNCH PLUS	£79.00

FILE MANAGEMENT
asmTREE	£239.00
B-Tree with Source	£120.00
BTree	£110.00
AXOS	£360.00
Btrieve v.5.01	£160.00
Btrieve OS/2	£360.00
Btrieve Multitasking	£360.00
Btrieve/N v.5.01	£360.00
Btrieve XENIX	£360.00
C English Database Interface	£120.00
C Index+	£175.00
c-tree	£230.00
Code Base 4	£89.00
db_Files	£239.00
d-tree	£299.00
dBC III	£160.00
dBC III Plus	£440.00
Emerald Bay C Dev. Toolkit	£299.00
RealDL/I	£599.00
WKS Library	£119.00

GENERAL
ASMLIB	£89.00
C Tools Plus	£79.00
C Utility	£120.00
C*Lib C Function Library	£120.00
Clipper toolbox	£239.00
DB/RA3	£125.00
dBase Program Utility	£55.00
dBase Program Utility Vol 2	£55.00
dBase Tools For C	£55.00
Finally	£60.00
Functions with source	£125.00
Integrated Dev. Libraries	£90.00
Pascal Toolkit	£90.00
Pascal Tools 1	£75.00
QuickPak Professional	£90.00
Quick Tools Library	£79.00
Spindrift Library	£90.00
Super Functions	£160.00
Tom Rettig Help Combo	£75.00
Tom Rettig Library	£60.00
Turbo C TOOLS	£79.00
Turbo Plus	£60.00
Turbo PowerTools Plus	£79.00
Turbo Functions	£69.00

GRAPHICS
dBase Tools for C Graphic	£55.00
dGE	£125.00
EGA Toolkit	£149.00
Essential Graphics	£180.00
Flipper	£120.00
FontWINDOW/Plus	£170.00
GEOGRAPH Professional	£239.00
Grafmatic CGA & EGA	£135.00
GraphiC	£239.00
GSS Graphics Dev. Toolkit	£299.00
GSS Kernel	£299.00
GSS Metafile Interpreter	£179.00
HALO-88	£175.00
HOOPS	£299.00
HOOPS 32	£349.00
MetaWINDOW/Plus	£169.00
scLASERplus	£299.00
SilverPaint	£60.00
SilverPak	£179.00
Slate	£215.00
Universal Graphics Library	£130.00

OP. SYSTEM ENHANCEMENTS
C Sharp	£300.00
DESQview API C Lib/Ref	£125.00
GEM Toolkit	£299.00
GEM/3Programmer's Tlkt	£299.00
Multi C	£89.00
Multi Dos Plus	£60.00
OS/2 Programmer's Toolkit	£215.00
OS/286	£299.00
OS/386	£299.00
Resident C	£120.00
Sapiens V8	£180.00
Security Lib with Source	£160.00
Synergy Toolkit	£239.00
Time Slicer	£179.00
VMFM	£115.00
Wendin OS Toolbox	£60.00
Wendin PC UNIX	£60.00
Wendin PC VMS	£60.00
Windows Development Kit	£295.00

QUERY
CBTREE/SQL	£239.00
CQL Query System	£239.00
dQUERY/Lib	£179.00
db_RETRIEVE	£239.00
XQL	£480.00
Xtrieve PLUS	£360.00
Netware SQL	£385.00

REPORT
CB/IQ Report	£139.00
Coreports	£179.00
Focus Report Writer	£210.00
r-tree	£179.00
R&R Relational Reportwriter	£89.00
R&R Clipper Module	£45.00

TEXT SCREENS
Animator COBOL	£120.00
Application Manager	£450.00
C-Worthy	£160.00
C-Worthy + Forms	£240.00
C-Worthy + Forms + Source	£395.00
C-Worthy All C compilers	£480.00
C-Worthy Forms only	£80.00
C-Worthy Source code only	£155.00
C-Worthy OS/2	£240.00
C-Worthy + Forms for OS/2	£320.00
C-Worthy + Forms + Source	£480.00
C-Worthy OS/2 & MS-DOS	£315.00
C-Worthy + source	£795.00
Curses	£150.00
Curses Window Dev. Kit	£75.00
Data & Windows	£149.00
DataWindow	£179.00
dWindows for db3	£60.00
Face-it	£60.00
Flash-up w/Toolbox	£85.00
Forms II	£179.00
HI-SCREEN XL	£89.00
Multi Windows	£179.00
On-Line Help	£90.00
PANEL PLUS	£299.00
PC Forms	£90.00
Pop Screen	£55.00
POWER SCREEN	£79.00
Pro Screen	£60.00
SCREENIO	£245.00
Screen Sculptor	£75.00
ScreenStar	£120.00
Screenplay	£110.00
SoftCode	£120.00
SoftCode Template	£30.00
Softscreen/Help Source	£150.00
Sourcewriter	£605.00
Terminal Control	£605.00
Turbo Magic	£60.00
UI Programmer	£179.00
V Screen	£39.00
VCScreen	£90.00
Vitamin C	£135.00
VLib with Source	£90.00
Windows for C	£120.00
Windows for Data	£179.00
Z View	£149.00

DEBUGGERS & DISASSEMBLERS
386 Debug	£120.00
Advanced Trace-86	£110.00
Advanced Disassembler	£180.00
Breakout	£75.00
Codesmith	£89.00
D'Scope 2	£149.00
dB Scope	£49.00
dBug for dBase	£120.00
DIS*DOC	£79.00
Periscope I	£480.00
Periscope II	£110.00
Periscope II-X	£89.00
Periscope III	£845.00
Periscope III Plus	£1,210.00
Serial Test	£299.00
Soft Probe II/TX	£239.00
Soft-X-plore	£79.00
Sourcer + BIOS Pre-Processor	£89.00

DOCUMENTORS
ASM Flow	£60.00
Clear+ for dBASE	£125.00
Clear+ for C	£125.00
dAction	£55.00
dAnalyst	£139.00
Design/2.0	£215.00
dFLOW	£89.00
Documenter	£179.00
Flow Charting II Plus	£139.00
FoxToolBox	£179.00
PolyDoc	£115.00
PolyXREF (All languages)	£60.00

EDITORS
BRIEF v.3.0 *NEW*	£149.95
BRIEF OS/2 (U.K. Version)	£149.95
dBRIEF v.2.1 (U.K. Version)	£79.95
BRIEF with dBRIEF	£219.95
C-EDIT	£239.00
CSE with C Source	£60.00
EMACS	£199.00
Epsilon	£110.00
ME Editor with C Source	£115.00
Micro ISPF	£110.00
Norton Editor	£49.00
PC/EDIT Plus	£179.00
Personal Rexx	£79.00
PI Editor	£119.00
SPF/PC	£149.00
Turbo Basic Editor Toolbox	£59.00
Vedit+	£115.00

HELP GUIDES AND TRAINING
dBASE on-line	£60.00
Norton Guides for ASM	£75.00
Norton Guides for BASIC	£75.00
Norton Guides for C	£60.00
Norton Guides for OS/2	£89.00
Norton Guides for Pascal	£75.00
Programmer's Library	£239.00

LINKERS & LIBRARIANS
Archive Link	£239.00
Link & Locate ++	£245.00
LinkLok	£245.00
PLINK86 Plus *Now Polytron*	£299.00
PolyLibrarian I	£60.00
PolyLibrarian II	£85.00
.RTLink	£180.00

MAKE UTILITIES
Advantage Make	£79.00
BSW-Make	£59.00
PolyMake v.3.1 *NEW*	£85.00
Q Make	£60.00
Seidel Make Utility	£60.00

PROFILERS & OPTIMISERS
Codesifter	£75.00
DOSTRACE	£59.00
PC Metric	£120.00

PROTOTYPERS & GENERATORS
Automated Programmer	£605.00
Cogen	£275.00
COMPEDITOR	£110.00
Dan Bricklin Demo II	£120.00
Data Boss for C	£245.00
Data Boss for Pascal	£245.00
Genifer	£239.00
Instant Replay III	£89.00
Matrix Layout	£89.00
Sycero db (U.K.)	£595.00
Sycero db Net (U.K.)	£745.00

SYNTAX ANALYSERS & LINTS
PC Lint v.3.0	£75.00
Pre-C	£189.00

TEXT & UNIX TOOLS
C Programmers Combination	£79.00
C ToolSet	£59.00
MKS LEX & YACC	£149.00
MKS Toolkit v.3.1 *NEW*	£149.00
MKS Toolkit v.3.1 *UPGRADE*	£50.00
PC Fortran Tools	£99.00
PC Yacc	£239.00
PolyAWK	£60.00
PolyAWK OS/2	£115.00
QPARSER Plus	£180.00
Text Management Utilities	£75.00
Turbo Professional	£75.00

TRANSLATORS
BAS_C Commercial	£229.00
Bastoc	£480.00
dBx dBASE to C v.4.0	£345.00
dBx Library Source	£250.00
dBx UNIX/XENIX Source	£470.00
FOR_C v.2.0 MS-DOS *NEW*	£565.00
FOR_C v.2.0 XENIX *NEW*	£630.00
FOR_C++ XENIX/386 *NEW*	£1195.00

VERSION CONTROL
PVCS Personal v.2.1c	£85.00
PVCS Professional v.3.1	£225.00
PVCS Network (5 User)	£735.00
PVCS/PolyMake OS/2	£CALL
PVCS/PolyMake VAX/VMS	£CALL

BACKUP - RECOVERY - REPAIR
dSalvage	£59.00
Fastback	£105.00
Fastback Plus	£115.00
Mace Utilities	£59.00
Norton Utilities Advanced	£89.00
PC Tools Deluxe	£49.00
XTree Net	£235.00

CACHE
PolyBoost II	£49.95

COMMS. & TERM. EMULATORS
Blast Communications	£149.00
Carbon Copy Express	£199.00
Carbon Copy Plus	£115.00
Crosstalk Mark 4	£146.00
Crosstalk XVI	£115.00
Mirror II	£45.00
PC Anywhere	£55.00
PC Anywhere III	£85.00
Relay Gold	£149.00
Relay Silver	£89.00
Sidetalk II	£70.00
SmarTerm	£75.00
SmarTerm 125	£175.00
SmarTerm 220	£115.00
SmarTerm 2392	£135.00
SmarTerm 240	£199.00
SmarTerm 400	£99.00
SmarTerm 4014	£135.00
VTek	£89.00
VTerm 220	£145.00
V Term 4010	£149.00
Vterm III	£115.00
Z Stem PC-VT100	£59.00
Z Stem PC-VT220	£89.00
Z Stem PC-VT240	£175.00
Z Stem PC-VT4014	£59.00
Zap Term Emulator	£59.00

FAX OR TELEPHONE CREDIT CARDS ORDERS VISA OR ACCESS

PRICING
All prices exclude V.A.T., but include first class recorded delivery.

EDUCATIONAL DISCOUNTS
Generous Educational discounts available on certain products.

CATALOGUE & FACT SHEETS
Call for our new Autumn '89 Catalogue, 58 pages of Software Development Tools.

THE SOFTWARE CONSTRUCTION CO. LTD.
THE RED HOUSE, 84 HIGH ST., BUNTINGFORD, HERTS. SG9 9DJ (0763) 73455 FAX. (0763) 73460

CIRCLE NO. 704

Upping the pace

Faster baud rates are being used more often because of the introduction of modems with data compression as an in-built function. This means that you can use a modem at (say) 2400 baud but, because the incoming data was compressed, your computer needs to be able to receive data much faster. To get the best from a V22bis modem with MNP5, you must connect to it at 4800 baud or above. A V32bis 9600 baud modem, with data compression, needs to be run at 19200 baud or greater. We have seen that a slow CPU has less chance of being able to service such a high-speed modem. Unfortunately, it doesn't necessarily follow that a fast CPU will, by itself, guarantee modem compatibility.

Communicating at 9600 baud means that 960 characters are coming in to the serial port every second. Anything that disables interrupts for longer than a millisecond will cause lost characters, because the interrupt service routine will not get a chance to pick up the data. For instance, the EGA BIOS can disable interrupts for up to 1/60th second when changing screen colours. At 9600 baud up to 16 characters might be lost.

Similar problems occur when multitasking operating systems, such as DESQview and CDOS (Concurrent DOS), are used. If your character-reading software loses access to the CPU for any length of time, you must inevitably lose data. A multitasking operating system is bound to control CPU access rights. Such tasks as switching expanded/extended memory pages in and out of context are usually done with interrupts disabled and, once again, characters get lost in the process. Error correction in the modem cannot help, as the data is being lost between the modem and the CPU. Error correction is useful only when the problem lies between the remote machine and the modem. Faster CPUs and UARTs will not necessarily resolve this position either.

```
;
; see whether a char is available
;
;      memory accesses
;      clock periods
;
reread:
    mov    dx,tps                ; 10  1
    in     al,dx                 ; 10  1
    and    al,00000001b          ;  4
    jz     exserv                ;  4
;
; if no character, exit
; if character, read it
;
    sub    dx,5                  ;  5
    in     al,dx                 ; 10  1
;
; store in circular buffer
;
    mov    bx,offset inbuf       ;  4
    mov    si,nextin             ; 14  1
    mov    [bx+si],al            ; 17  1
    inc    si                    ;  3
    and    si,2047               ;  4
    mov    nextin,si             ; 15  1
;
; now see if fifo buffer is empty
;
    jmp    reread                ; 15
;
;   instructions             = 13
;   clocks                   =    116
;   memory accesses          =         5
;
```

Figure 4 – ISR amended for 16550 FIFO

It's a small price to pay for a chip that is more effective than quadrupling your CPU speed

The answer to the problem lies in the 16550 UART. The specification for this device is similar to the 16450, but it has the added feature of 16-byte FIFO (First In First Out) buffers. To explain: both the 8250A and the 16450 have a single receive holding register. Each incoming character is transferred to this register as soon as the stop bit is detected, overwriting the register's previous contents. If you had not yet read the character already in the holding register, then tough trousers. All you now have to show for it is an overrun error.

The 16550 will emulate this 8250A mode if you desire. But the whole point of fitting one is to make use of its FIFO buffered mode. The 16550 has 16 holding registers, filled up on a circular first-in-first-out basis. This means that if, for instance, the EGA BIOS disables interrupts for 1/60th second then, with data coming in at 9600 baud, you won't lose a single bit.

To make the interrupt service routine from Figure 2 work with the 16550 FIFO buffers, apply the amendments given in Figure 4. These are straightforward. We read the status of the UART, checking for a character remaining in the FIFO in the same way that the previous routine checked the holding register. The only difference is the `reread` loop, which caters for the possibility of reading multiple characters at one interrupt. The FIFO is simply treated as a holding register with space for up to 16 bytes.

Back to our timing calculations. Using the same assumptions as before, that gave an overhead per character of 550 clock periods for an 8250A, the overhead per *additional* character, taken from the 16550's FIFO and stored in memory, is only about 150 clock periods. Were the FIFO to be completely full, it would take 2800 clock cycles to empty it. If we look at Figure 3 again, we see that an 8250A interrupt service routine, collecting one byte at 2400 baud, takes about 2500 clock periods. We have a fair chance of picking up *sixteen* bytes in the same period of time with the 16550 – which is 38400 baud.

In real life, I have found that a 4.77 MHz IBM XT, which, with its original 8250A UART, had difficulty keeping up as a terminal emulator at 9600 baud, could breeze along at 38400 baud with a 16550 fitted instead. Even a 386 machine, which lost characters when task-switching on EGA screens under CDOS, started to behave perfectly once a 16550 was fitted and properly enabled. This ability of the 16550, to overcome the limits of real time communications when combined with multitasking operating systems, is presumably why IBM added FIFO support for the 16550 in their Advanced BIOS.

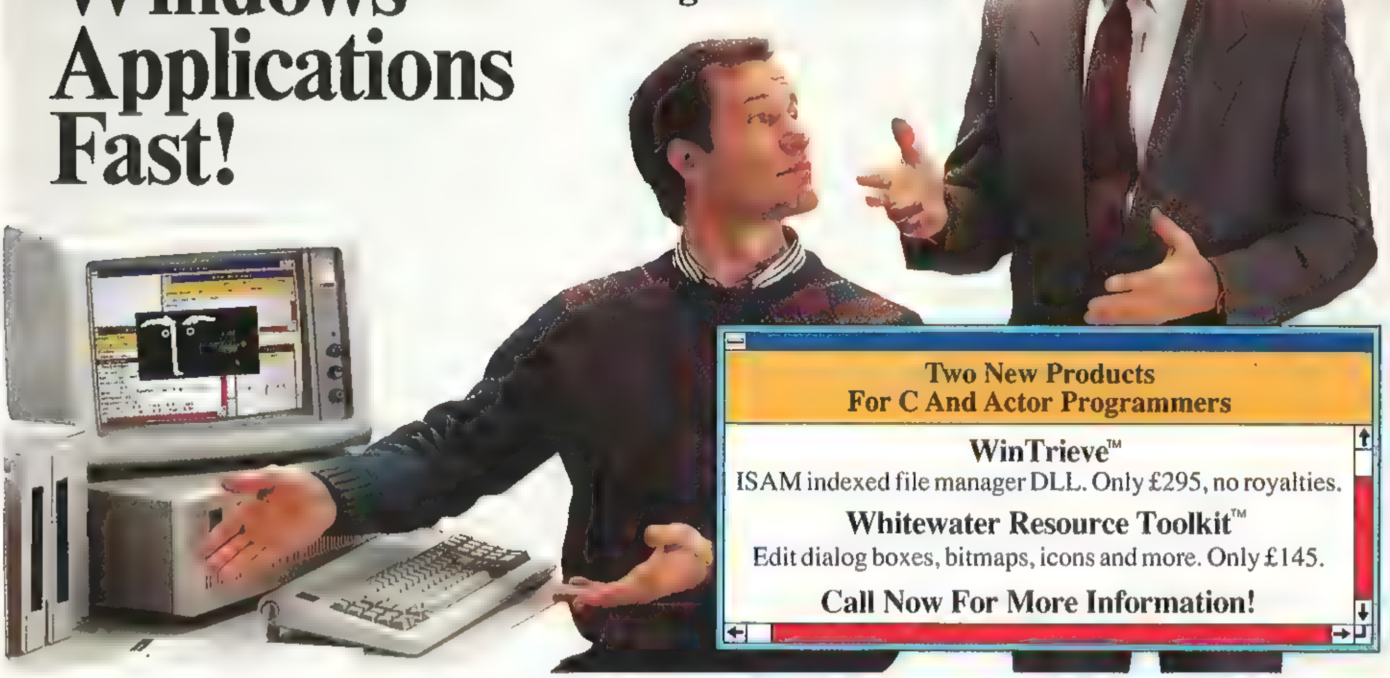
Inside the 16550

Figure 5 illustrates the register structure of all three UARTs. If you've programmed 8250A UARTs before then it will make perfect sense. If you haven't, you will need to swot up before attempting the next paragraph.

The 16550 FIFOs are enabled by writing a word, with bit 0 set, to the same register that you would normally read to find out the source of a UART interrupt. Setting bits 1 and 2 resets the contents of the receive and transmit FIFOs respectively, while setting bit 3 enables DMA multibyte transfers to and from the FIFO buffers. Bits 6 and 7 enable a trigger to be set for the receiver FIFO interrupt at either 1, 4, 8 or 14 byte levels. For instance, if a trigger level of 8 is set, then no receive character interrupt will be generated while the FIFO contains less than eight characters. When the eight character limit is reached, the interrupt is generated. Additionally, a timeout interrupt will be generated if the FIFO contains unread characters and hasn't changed for a period equivalent to that taken to receive four characters at the current baud rate. This is indicated by the UART setting the (previously unused) bit 3 of the interrupt identification register, as well as the usual bit 2, which shows that a character is available.

Develop Microsoft® Windows Applications Fast!

"How Did You Get This Windows Application Done So Fast? "Everyone Else Is Still Reading the Manuals..."



The Windows Advantage

You've selected Microsoft Windows as your graphical user interface. Windows applications are easier for end-users to learn and use. It is **the** standard for PCs.

Now you want Windows applications completed as fast as possible. But it takes time for your programmers to get up to speed on the intricacies of Windows development.

Why The Delay For Windows Applications?

Programming for the Windows environment is very different from character-based programming. Just learning the 450 Windows function calls can take months. Getting used to its event-driven system and memory management takes additional time.

Programmers who are new to Windows development are looking for support to help them produce high-quality Windows programs quickly and efficiently.

3 Ways To Speed Up Windows Application Development

Now there are specific ways to speed up applications development under Windows.

- **Code reduction.** There is a fast, effective way to develop Windows applications. A way that cuts the amount of code needed. A way that by-passes much of the complexity of Windows programming. A way that doesn't compromise performance.
- **Short learning curve.** There is an easy, convenient way to get your entire programming staff up to speed with Windows within weeks.
- **From prototype to complete application.** There is a way to quickly develop prototypes of Windows applications in days, not months. Then, turn those into completed applications quickly.

1 Product, 3 Solutions

There is **one** product which brings **all three** solutions to your Windows programmers. It's called **Actor**®

Actor is a proven product. It's now being used in thousands of software development groups world-wide.

Actor Does Windows

Actor gives your programmers a fast, effective way to develop Windows applications. **Actor's** complete development environment allows you to produce everything from prototypes through stand-alone applications. Programs developed in **Actor** are fast and support all Windows features.

Actor offers interactive source code debugging and testing—greatly reducing development time. Your programmers benefit from high code reusability because **Actor** is an object-oriented programming language. This allows them to build on existing code, eliminating duplication of effort. They can even dynamically link to existing C code.

Actor also provides one more essential: fast, responsive support from NEOW - The Windows Specialists, UK experts in Actor and Windows programming.

If you are programming applications for Windows, you should be using **Actor**. Users report that **Actor** cuts Windows programming time by at least 50%.

Act Now!

See for yourself how **Actor** can help your Windows programmers! The first step is to order our free booklet, **For Faster, Better, Smaller, Smarter Windows Programming.** Examine detailed information about using **Actor,** complete with actual source code examples.

Call us at 06286 68334. We'll send you the booklet and complete details about **Actor.**

Take 60 seconds now to save months of development time in the future.

NEOW **- The Windows Specialists**
Unit 12, Progress Business Centre,
Whittle Parkway, Slough. SL1 6DQ.
Telephone: 06286 68334. Fax: 06286 65722

CIRCLE NO. 705

Figure 5 – Summary of Registers for 8250/16450/16550

	Base	Base	+1	+2	+2	+3	+4	+5	+6	+7
	RX Data DLAB=0	TX Data DLAB=0	Interrup Enable	Interrup Identity	FIFO Control	Line Control	Modem Control	Line Status	Modem Status	Scratch Reg
BIT	R/O	W/O	R/W	R/O	W/O	R/W	R/W	R/W	R/W	R/W
0	Data 0	Data 0	Receive Data Ready	0 when interrup pending	FIFO Enable	Word Length Bit 0	DTR	Receive Data Ready	Delta CTS	
1	Data 1	Data 1	Transmit Hold Reg Empty	Interrup ID Bit 0	RX FIFO Reset	Word Length Bit 1	RTS	Overrun Error	Delta DSR	
2	Data 2	Data 2	Receive Line Status	Interrup ID Bit 1	TX FIFO Reset	Stop Bits	Out 1	Parity Error	Trailing Edge RI	
3	Data 3	Data 3	Modem Status	16550 ONLY Bit 2	DMA Mode Select	Parity Enable	Out 2 IBM Intr En	Framing Error	Delta DCD	
4	Data 4	Data 4	0	0	Reserved	Even Parity	Loop	Break Detect	CTS Status	
5	Data 5	Data 5	0	0	Reserved	Stick Parity	0	Transmit Hold Reg Empty	DSR Status	
6	Data 6	Data 6	0	16550 FIFO Enabled	RX Trigger LSB	Break Set	0	Transmit Shift Empty	RI Status	
7	Data 7	Data 7	0	16550 FIFO Enabled	RX Trigger MSB	DLAB	0	16550 RX FIFO Error	DCD Status	
		R/W	R/W							
		DLAB=1 Divisor LSB	DLAB=1 Divisor MSB							

Note: NatSemi recommends that the Line Status Register NOT be written to, as this is only supposed to be done in factory testing. The only 8250A/16450 bits that change in 16550 FIFO mode are bits 3, 6 and 7 in the interrupt identification register and bit 7 in the line status register.

For an IBM PC, with a 16550 on COM1 at base address 03F8H, the FIFO mode is enabled by writing a binary 1 to 03FAH. Thereafter the UART will operate in FIFO mode until the FIFOs are disabled, by writing binary 0 to 03FAH, or it is powered down (FIFO mode is off by default). For a great deal of software, enabling FIFO mode in this way will result in an increase in performance – even though the software is not aware of the 16550. Any programs which reread the UART status register after servicing an interrupt (so that they can check to see if anything else needs action) will make full use of the 16550 FIFOs. There are, however, some notable incompatibilities that you ought to watch out for:

- It is common for an initial check to be made for the existence of a particular port by reading the interrupt identification register and masking off the low nibble. Reading an unused port on Intel CPUs gives a value of 0F0H. If an 8250A is at the address, it returns a value of zero. If a 16550 is present, with FIFO mode enabled, then the value returned is 0C0H. Since this is a non-zero result, the program might (erroneously) conclude that there is no serial port present. The IBM ROM BIOS does precisely this. If you enable FIFOs on a 16550 and then reboot, you'll find that the BIOS no longer recognises the 16550 as a valid serial port.
- The interrupt identification register on the 8250A can be read and used as an offset to a jump table without having to mask out any bits – the unused ones all return clear. On a 16550 with FIFOs enabled, bits 6 and 7 must be masked out, as they will both return set. Bit 3 also needs to be masked out, because, as I mentioned above, it is used to indicate a timeout interrupt by the 16550 in trigger mode. To crash your machine fast, try enabling FIFOs, then running software that uses the unmasked interrupt identification register value as a jump table offset.
- In a similar way, some software might read the line status register and use that as an offset into a jump table. An error in the receiver FIFO will cause unpredictable results because the line status register, which always returns with bit 7 clear on the 8250A, will return with bit 7 set. Note that the effect is more subtle. The software may work with FIFO mode switched on for some time, until eventually nemesis arrives in the form of a receive FIFO error.

If you are writing your own 16550 software, the only thing you need to do, apart from enabling the FIFOs, is force a reread of the line status register after each character is received. This ensures that the FIFO is emptied. The code in Figure 4 illustrates this point.

Brain Damage

No information on any chip is complete without an account of where the brain-damaged areas might be found. In the case of the 16550, it occurs in the transmit FIFO logic. One of the major weaknesses in the 8250A has always been its lack of any transmit flow control. This has meant that either the state of CTS (or some other signal) has to be sampled before every character sent, or, equally unsatisfactorily, a special interrupt routine, to handle modem status changes, has to be hooked into a transmit interrupt routine. This weakness is compounded in the 16550. If transmit FIFO logic is used, up to 16 bytes can be transmitted with no possibility of checking for hardware handshaking. Suppose that your error-checking modem hits a burst of line noise and tells you to stop sending to it while it recovers. The 16550 might not respond until 16 bytes later. While it's irritating to find that a feature is unusable, character transmission has never really caused as many problems as character reception, as it is always under your control anyway.

The price of the 16550 UART is about £11.00. If you suffer from lost data, either through running your communications sessions under multi-tasking operating systems, or through having a modem that stretches the capabilities of your system a little too far, it's a small price to pay for a component that can do more to solve the problem than quadrupling your CPU speed. You'll almost certainly have to order the part specially (make sure you order the DIL dual-in-line package), but, once it arrives, you simply plug it into the same socket that the previous chip was using. If you are thinking of buying a modem card or an extra serial board, make sure that either it has a 16550 on board or else a socketed (not soldered) UART (for easy replacement). Watch out for the 16552 DUART. This is a recently announced dual-channel version of the 16550, with identical software capability, but considerably faster than any of the earlier UARTs. It boasts read and write cycle times of 84 ns, compared to 755 ns on the 8250, and can, in theory, run at speeds of up to 1.5 Mbaud. Unlike the 16550, however, it isn't a simple plug-compatible replacement for an 8250. EXE

Andrew Margolis does a number of jobs at Margolis & Co, a software house and computer consultancy which specialises in communications and other low-level activities, and which also publishes the COMM+ Communications Processor.

Acknowledgements to Tim Ward of Mayze/Dowty, who provided the 16550 UARTs and the relevant data sheets, and who also are fitting the 16550 as standard in their new range of Quattro modems.

Further information is available from the National Semiconductor Corporation, The Maple, Kembrey Park, Swindon, Wiltshire SN2 6UT. Tel 0793 614141. The company prints a range of data sheets, describing the UARTs discussed in this article.

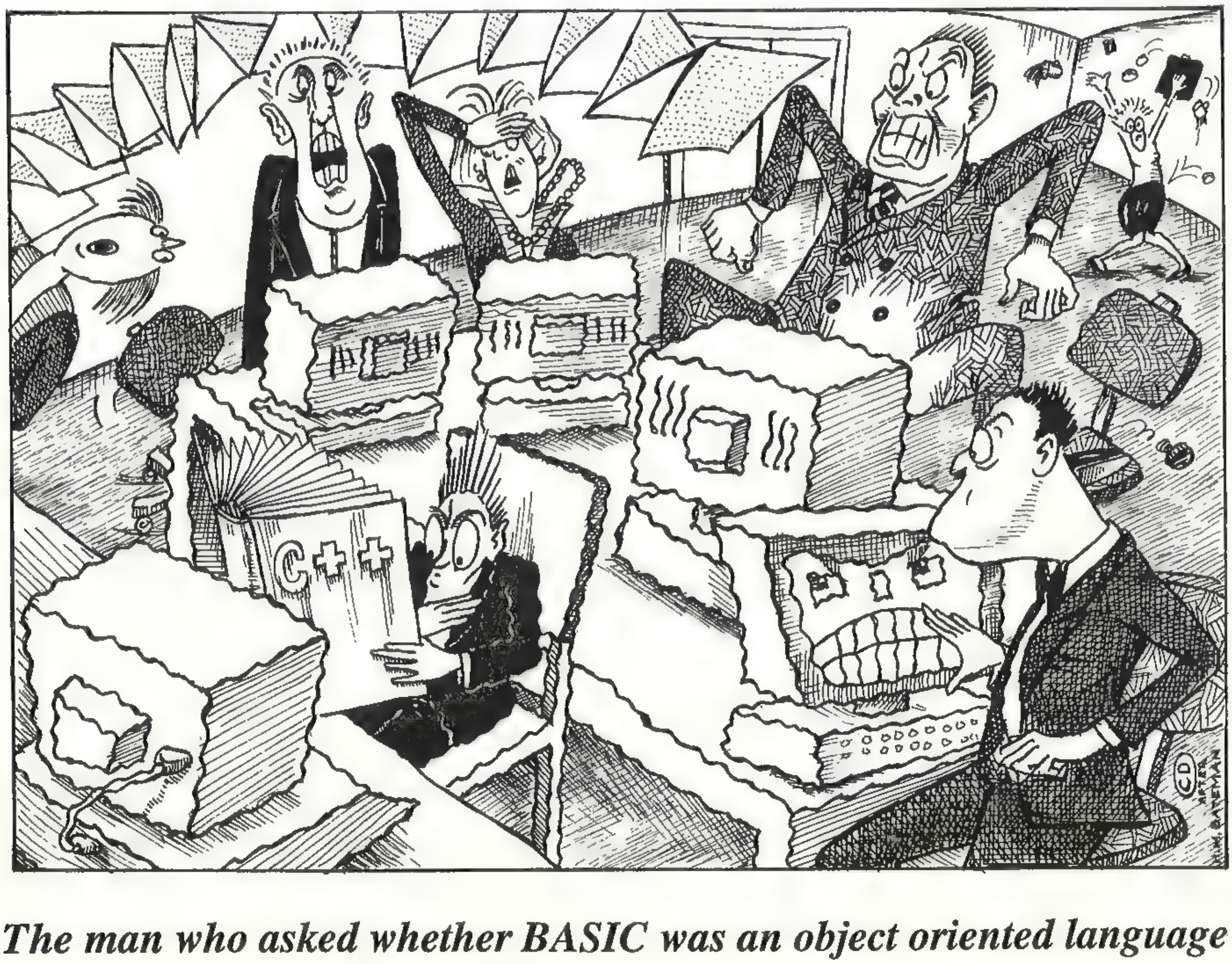
What can C++ do for you?

Paul Smith provides an overview of C++, the 'Super C' that's already changing the way that many people use the world's favourite language.

C++ was designed, says Bjarne Stroustrup, 'to make programming more enjoyable for the serious programmer'. C++ extends standard C in a number of ways, many of which are designed to support object oriented programming. A summary of the most important of these features follows.

The C++ language builds on the features of ANSI C, retaining a very high degree of compatibility. However, the purposes of the two languages are different: ANSI C serves to codify the existing C language and resolve inconsistencies between implementations. C++ is a new tool, intended to provide a new and expressive programming model. Where there are incompatibilities between the C subset of C++ and ANSI C, they exist intentionally as the price of important C++ features and concepts.

The high degree of compatibility between C++ and C gives a number of advantages,



The man who asked whether BASIC was an object oriented language

the primary ones being that: programmers can learn C++ very easily; the existing body of C software, which is very large indeed, can be interfaced to with very little effort; all the ANSI C libraries can be used directly from C++ programs.

Of central importance is the concept of user defined types. In standard C, it is easy to define a structure that combines data fields in a named entity. However, one cannot be very expressive in the manner in which one deals with this entity. C++ allows one to define entirely new data types, which may be used with all of the expressiveness of the standard C built-in types. For example, one can define a new fixed-point fractional number data type 'fixed' so that ordinary variables can be defined thus:

```
fixed c, d;
int a = 60;
```

and then perform ordinary arithmetic on them:

```
c = 647.825;
d = c * a;
```

The C++ language allows the programmer to define and work with user-defined types. They are identified with the keyword 'class' (C-style 'struct' types are also supported), as illustrated in Figure 1. User defined types are not merely the data that they represent. All the properties of the data that are associated with a user defined type (the information required to represent it, and the functions and operators that are necessary to initialise, manipulate, and dispose of it) are bound to that type.

Classes and Inheritance

C++ classes can contain data fields, which correspond to the fields in C structures. They can also contain functions and operators. Together, the members of a class represent the properties of that class. Members can be public, protected, or private. Public members can be accessed by any code, in the same way as standard C structure fields. Private members are only accessible by other members of the same class. Protected members are like private members, except that they may also be accessed by members of classes derived from the class in which they are defined. Members of 'class' types are private by default, and members of 'struct' types are public by default.

A new class can inherit the properties of other classes. If a class is derived from another class, it inherits all the properties of that class, except for those that are explicitly overridden by the programmer. This allows you to define hierarchies of classes that share common properties. New data fields, and new functions, only need to be defined where the new classes differ from their ancestors.

Figure 1 – A C++ Class Declaration

```
class complex {
        double  re, im;  // data fields holding complex number data public:
        complex();        // default constructor: re=im=0
       complex(double r = 0, double i); // constructor with arguments

        friend  double  abs(complex);     // function members
       friend  double  norm(complex);
        // ... more functions ...
        friend  complex sqrt(complex);

        friend  complex operator+(complex, complex); // operators
        friend  complex operator-(complex, complex);
        // ... more operators ...
        void    operator/=(complex, complex);
};
```

Figure 2 – Sample code for use of the new complex type

```
complex complex::operator+(complex a1, complex a2) // addition operator {
        return complex(a1.re+a2.re, a1.im+a2.im);

        // Creates a new 'complex' object, calls the constructor
        // complex(double,double) P that takes arguments - and
        // then returns a reference to the new object. }
```

Figure 3 – C++ provides default arguments

```
void goto_position(int column = 0, int row = 0)
{
        // call graphics handler to position at (column, row)
        // .....
};

// examples of usage:

        goto_position;              // moves to position (0, 0)
        goto_position(100, 120);    // moves to position (100, 120)
```

C++ classes are actually defined in two parts. First comes the class declaration, in the form shown in Figure 1. Second are the functions themselves, which are written in almost the same way as ordinary C functions. See Figure 2 for an example of code for one of the functions declared in Figure 1. Note the header of the function, which specifies the class to which the function belongs. Class objects may be allocated automatically, statically or dynamically, just like any other C++ variable. Special 'new' and 'delete' operators are defined to create and dispose of dynamic class objects.

Constructors

C++ allows the programmer to ensure that classes are always created in a known state. The programmer can, if necessary, define special functions called constructors to initialise objects. Constructors are just like ordinary functions – they have the same name as the class to which they belong, and they may be defined with or without parameters. A class may be defined with no constructors, or with one or both types of constructor. If a constructor has been defined, it is called every time an object belonging to its class is declared or created. Constructors without parameters are intended for the default case, and are also called to initialise elements of arrays; constructors with parameters are called when, at the time a class object is created, the parameters are specified.

Destructors are functions with the same name as the class, but prefixed by the tilde character. A destructor is called when an automatic or static class object goes out of scope, or when a dynamic object is disposed of. If members of a group of objects have common properties, but actually work in different ways, it is convenient to be able to write code that calls on those common properties the same way without knowing of the differences. For instance, similar objects may be stored in a linked list in memory, or accessed via a vector of pointers. The code that manipulates them might need to ask them to perform certain operations without knowing the actual type of each object. C++ allows one to call on objects of different classes, not knowing anything more than the common ancestor they are derived from. At run time, C++ will determine which object type is actually being called, and ensure that the correct function code is used to respond to the call. The one restriction is that the objects must derive from a common ancestor that shares all the properties called upon.

> [illegible] the future. I can't be locked into a particular [illegible]

TODAY

APPLICATION DEVELOPMENT ENVIRONMENT

Interoperability

[illegible] and data structures.

Environmental Support

[illegible] workstation, terminal or PC for targets that include mid-range and mainframe on-line [illegible] and PC networks. You can access data in files or relational tables with conventional I/O [illegible] applications across UNIX, XENIX, MS-DOS and proprietary operating systems including VMS, AIX, PRIMOS, and MPE without recompiling.

Data Independence

TODAY lets you build applications using relational databases like Oracle, Informix, or [illegible] a combination of both. You get the benefit of TODAY's full development functionality plus the freedom to choose the data structure best suited to your needs.

Please send me more information about TODAY

Name ______________________

Title ______________________

Company ______________________

Address ______________________

______________ Tel ______________

.EXE 11/89

TODAY APPLICATION DEVELOPMENT ENVIRONMENT

Computer Power Europe, Berkeley House, 51-53 High Street, Redhill, Surrey, U.K. RH1 1RX.
Telephone: 0737-764500. Fax: 0737-764525

COMPUTER POWER EUROPE

CIRCLE NO. 739

When a derived class overrides a function, and the programmer wants C++ to determine at run-time which function to call according to the actual class it belongs to, the programmer must have declared the function in the common ancestor class to be 'virtual'. This tells the compiler that it must use a table lookup mechanism to determine the actual function to be called at run-time. (In previous articles in .EXE, I looked at how different versions of Apple's Object Pascal implement virtual functions. The two mechanisms used are flat virtual function tables, in which there is a fixed (and small) run-time overhead on each virtual function call, and inverted virtual function tables which use substantially less memory at the cost of an iterative search through the table for each virtual function call. C++ uses flat virtual function tables).

C++ allows, as part of a class definition, standard C++ operators to be redefined to work in ways appropriate to the class. For example, although standard C arithmetic operators will work fine with ints, longs, doubles and so on, they would be confused by a user defined complex number type that stores numbers in two parts. In this case, the arithmetic operators can be overloaded within the class definition to tell C++ how to manipulate the class variables in arithmetic expressions. This is similar to a feature of Algol68, but is much more powerful. If you need to, you can overload the assignment operator, to ensure that classes that store information dynamically can be copied correctly. You can overload the →operator if dynamic objects are not stored in simple heaps. You can overload the array operator [ ]. In fact, all the C++ operators can be overloaded. You cannot, however, change the precedence of operators from their normal order of evaluation.

Ordinary functions can be overloaded, as well. There can be multiple versions of a function, all having the same name but accepting different arguments (the return types do not need to differ). The C++ compiler determines which function to call by examining the arguments.

Constructors can be used to control how memory is allocated for objects. Overloaded operators can be used to redefine how the objects are accessed and manipulated. For instance, you can overload the 'new' operator and implement a customised heap management system. The programmer has total control over memory management.

Where there are incompatibilities between the C subset of C++ and ANSI C, they exist intentionally

Prototyping

ANSI C introduced extensive facilities for strict function prototyping and type checking. These features originated in C++. C++ insists on functions being used for all operations connected with objects. To allow the programmer to control this potentially inefficient way of working, functions can be defined as being 'inline'. At the compiler's discretion, inline functions are embedded in the program code to avoid the function call overhead. This can lead to significant performance improvements. The C++ programmer can specify default arguments for functions. Default arguments can be omitted when the function is called, and the compiler will fill in the correct values. See Figure 3 for an example.

In C, one can refer indirectly to variables by passing around their addresses. C++ abstracts this into the concept of 'reference types'. A reference type is an alias for the object it references. It can be used interchangeably with an ordinary variable; the C++ compiler automatically generates all the code necessary to de-reference it. C++ allows functions to return reference values. A reference type is declared using the '&' symbol thus:

```
double d;
double& e = d;
```

The reference variable `e` is now an alias for `d`, with which it can be used interchangeably. Another way in which C++ is different, compared to C, is the // comment delimiter. All characters after // on a line are treated as a comment.

What's New In 2?

The main new features in version 2.0 of C++ are summarised in Figure 4. The most important of these are support for multiple inheritance, and type-safe linkage. Multiple inheritance exists when a class inherits the properties of more than one class. Applying this technique, a class may be created that is a composite of base classes that represent independent concepts, as illustrated in Figure 5. Each base class is treated as a separate sub-class within the derived class. Public and protected base class members can be accessed just as if they were declared within the derived class, subject to a few restrictions which serve to eliminate ambiguities.

Many object oriented programming languages, such as Object Pascal, constrain the thinking of the program designer by only allowing single inheritance. C++ 2.0, by allowing multiple inheritance, removes this constraint. Single inheritance tends to force the program designer to create a single hierarchical tree of classes, all inheriting the properties of a single base class. This is usually quite effective, but it can be difficult to define a straightforward hierarchy that represents a very complex system: the designer has to work harder. It may be more convenient to break down a problem into smaller, independent hierarchies of classes that can be developed without knowledge of the properties of other classes. This is particularly true of general-purpose classes that are intended to be reused. Multiple inheritance allows these independent classes to be combined, exactly as required, for specific uses. Multiple

```
Multiple inheritance,
type-safe linkage,
better resolution of overloaded functions,
recursive definition of assignment and initialization,
better facilities for user-defined memory management,
abstract classes,
'static' member functions,
'const' member functions,
'protected' members,
overloading of operator -> ,
pointers to members.
```

Figure 4 – The main new features in C++ 2.0

```
class A { ... };
class B { ... };
class C : public A, public B { ... }; // C inherits A and B
class D { ... };
class E : public C, public D { ... }; // E inherits A, B, C, and D
```

Figure 5 – Illustration of multiple inheritance

inheritance can lead to ambiguities, and the C++ compiler detects ambiguous code and reports it as an error. For instance, it is fine for a new class to inherit the properties of two different classes that, in turn, inherit the properties of a common base class. The compiler will insist, however, that the base class member is qualified by the name of one of the sub-classes that contains it. Thus, C++ ensures that exactly the member that was intended is accessed. If the keyword 'virtual' is used to qualify common base classes, C++ will ensure that only one instance of the virtual base class exists in the derived class however many times it is derived. Otherwise, one instance will exist for each derivation of the base class.

As an extension of the mechanism that had to be implemented to allow the overloading of functions, C++ ensures that the arguments passed to external functions are compatible with their original definitions. The compiler does this by 'mangling' the function name, with its arguments, to create a compound name that contains all the necessary information. C++ considers the name of a function to be a combination of the original function name plus all its arguments: this is the name passed to the linker. Accordingly, if an external function is locally prototyped with an incorrect argument list, the linker will treat it as a different function and report an error.

Getting started

Bryan Boreham of Kewill Systems says 'it takes around three months for a good programmer to start producing useful code in C++. Knowledge of C is a very

C++ classes can contain data fields, which correspond to the fields in C structures

small advantage.' However, Bjarne Stroustrup told me that, in his experience, it takes new C++ programmers anything between two days and a year to start doing useful work with C++. Two days for experienced C programmers that don't need to use the object oriented programming features of C++, and up to a year for people who are new to both C++ and object oriented programming.

There are a large number of C++ 2.0 compilers becoming available. Zortech have already shipped their compiler. The AT&T implementation is available (or soon will be) for 80386s, for Sun workstations and for the Apple Mac.

Conclusion

C++ was conceived and developed by real people to solve real problems. It is incorrect to define C++ (as did Andy Redfern in Personal Computer World, recently) as a language that 'started life in academia as an exercise in improving C'. C++ is much more than that. It began as a solution to a genuine need, and throughout its life the development of C++ has been guided by pragmatic principles.

Paul G Smith is a freelance software development consultant and technical writer. He can be contacted on CIX as pgsmith, and on AppleLink as UK0310.

More maths matters

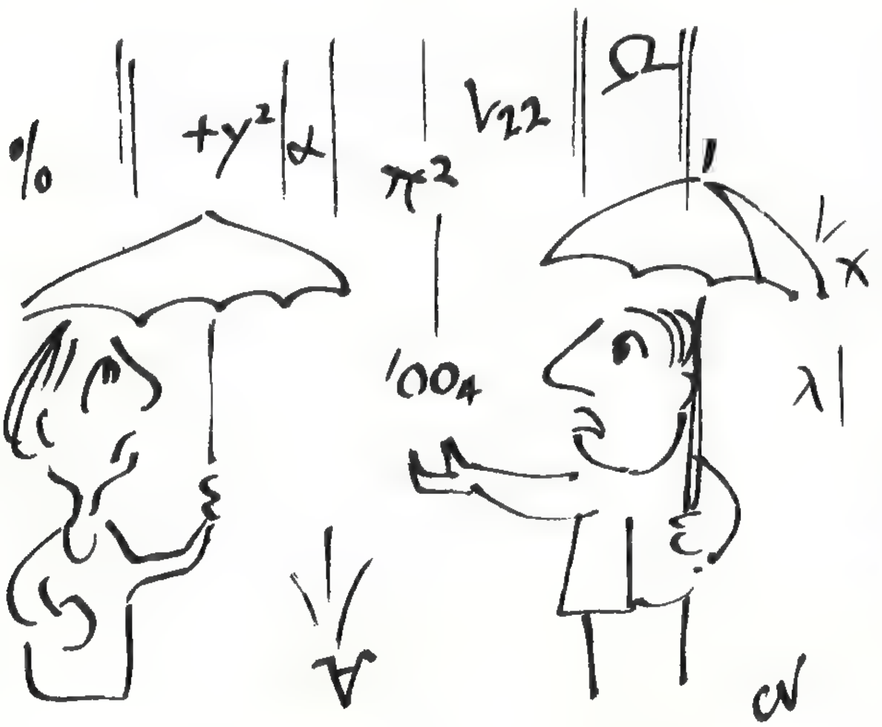
Darrel Ince's .EXE article about the formal methods of software development led many of you to write asking 'How is it done?' So we invited him back to explain.

In a previous article (.EXE, May '89) I explained how mathematical methods of software development are beginning to be used in British industry. That article concentrated on the advantages of these so-called formal methods of software development. This article takes a much closer look at the mathematics involved and gives an example of a specification expressed in mathematics.

Before wading in, a little background is necessary, describing a branch of mathematics known as set theory. This is the corpus of laws and theorems about collections of objects known as sets. A set is a very simple object, its distinguishing feature is that it doesn't contain duplicates, thus
{Thomas, Williams, Roberts}
is an example of a set of employees and
{react1, cracker2, distill1, distill2, distill3}
is an example of a set of chemical reactors. However,
{VAX1, VAX2, PC1, PC2, PC3, PC2}
is not an example of a set because it contains a duplicate entry (PC2). There is a special set, equivalent to zero in numerical mathematics, known as the empty set: a set with no members. It is usually written as { }. In a similar way that mathematics has operators such as +, – and *, set theory has a series of operators which can be applied to sets. The first is set union $\cup$. This forms a set from two sets. The result is a set which has the same members as the two sets to which $\cup$ is applied. For example, the result of
{23, 44, 67, 88, 90} $\cup$ {33, 44, 23, 89, 90}
is the set
{23, 44, 67, 88, 90, 33, 89}
Notice that 44, which is a member of both the sets to which $\cup$ is applied, only occurs once in the result set.

A useful operator is the set intersection operator $\cap$. This takes two sets and forms a set which contains only those elements which are common to the two. Thus,
{Ince, Watts, Williams, Tomkins} $\cap$ {Davis, Watts, Tomkins}
is equal to the set
{Watts, Tomkins}
since both sets have only two elements in common. Another commonly used operator in formal specifications is the set difference operator —. This removes the contents of one set from the other set. Thus,
{PC1,PC2,VAX1,VAX2} — {PC1,VAX2}
is equal to
{PC2,VAX1}

The operator $\in$, which is read as 'belongs to', takes an element of a set on the left hand side, and a set on the right hand side. It gives the value *true* if the element is contained in the set, and *false* otherwise. Thus,
Davis $\in$ {Roberts, Ince, Wilson, Davis}
is true and
Thomas $\in$ {Roberts, Ince, Wilson, Davis}
is false. A final operator is the **card** (ie cardinal number) operator. When applied to a set, it returns the number of elements in that set, thus,
card{1,5,78,89,67} = 5
card {} = 0



This short tutorial has described the base mathematics used for all formal methods of software development. To see what a formal specification looks like, we will consider a small problem.

The chemical plant

The problem that is to be specified is a small control system for a chemical plant. I shall use the formal method of software development called 'VDM' (Vienna Development Method) for the specification.

Figure 1-Natural language specification for chemical plant control system

```
A system is to be developed which controls the operation of a
number of chemical reactors. A series of commands are to be pro-
vided for plant operators. The SHUT_DOWN command shuts down a
specified reactor, the START_UP command starts the operation of a
specified reactor. The OPERATING command returns with the number
of reactors currently operating. No more than 100 reactors will be
controlled by the system, and no more than 20 of these reactors
will be operating at the same time.
```

Figure 2-Specification of stored data

```
react_set = set of reactors
system :: shut_down, operating: react_set
```

The natural language specification for the system is shown in Figure 1.

It is worth stressing that, because of space limitations and the tutorial nature of the article, the example I have used is unrealistically small. It could be specified using natural language without much loss of accuracy. Only when larger real-life systems are specified with mathematics do you get significant increases in clarity and exactness.

The first step is to identify stored data. In the example above, this is the collection of reactors which have been shut down and those which are operating. This stored data can be specified as shown in Figure 2. The first line states that a collection of reactors will be known in the specification as *react_set*. The second line states that the system will consist of two sets of reactors: *shut_down* (those reactors which are not currently working) and *operating* (those reactors which are).

The next stage is the derivation of the data invariant. This is a condition which must be true throughout the execution of a system. For example, in a banking application, the data invariant might state that no account is allowed to go beyond a particular overdraft limit.

The data invariant for the reactor system is: no more than 100 reactors will form part of the system, no more than 20 reactors will be working at the same time and a reactor will not simultaneously be shut down and operating. That is, if you take a snapshot of the system at any time during its operation, then you will find that the number of reactors will be less than or equal to 100, the number of operating reactors will be less than or equal to 20 and the intersection of the *operating* set and the *shut_down* set will contain no members. The data invariant can be written mathematically as shown in Figure 3.

The preamble inv-*system(s)* ≜ is just a way of saying *for any system s the data invariant is defined as*. The right hand side of the ≜ symbol specifies what the invariant is. Each part of the data invariant is separated by the symbol ∧ which can be read as *and*.

It is easy to see how the mathematical specification maps to the English version given above. The first line *shut_down(s)* ∩ *operating(s)* states that the intersection of the shut down reactors in the system and the operating reactors in the system will be the empty set, ie that a reactor cannot simultaneously be functioning and shut down. The second line states that the combined collection of shut down reactors and operating reactors will contain no more than 100 members. The final line states that there will never be more than 20 reactors in the set of operating reactors.

With very large systems, proof becomes so difficult that computer assistance is required

Operations

With the data now specified, the next stage is to identify any operations which must be supported by the system; these correspond to the functions of the system. In our reactor example, each operation will correspond to a command. These are:

- An operation which corresponds to the SHUT_DOWN command. This transforms an operating reactor to one which is shut down.
- An operation which corresponds to the START_UP command. This takes a reactor which is shut down and starts it operating.
- An operation which corresponds to the OPERATING command. This returns with the number of reactors which are operating.

Once identified, these operations can be specified mathematically. The specification for the SHUT_DOWN command is shown in Figure 4.

The first line gives the operation's name and parameter. SHUT_DOWN takes reactor *reac* – the reactor to be shut down. The next lines specifies the type of access the command requires to the stored data. There are two types of access: read access (objects in the data are read-only) and write access, in which objects are read and altered. In the case of SHUT_DOWN, write access is required to the system. This is signified by the keyword *wr*.

The next part of the specification defines the effect of the SHUT_DOWN command on the system s. This specification consists of a pre-condition and a post-condition, introduced by the keywords *pre* and *post* respectively.

The pre-condition states the condition under which the operation will be defined.

```
inv-system(s) ≜  shut_down(s) ∩ operating(s) = {}
                 card (shut_down(s) ∪ operating(s)) ≤100 ∧
                 card(operating(s)) ≤20
```

Figure 3-Specification of the data invariant

```
SHUT_DOWN (reac: reactors)
ext wr s: system
pre reac ∈ operating(s)
post    operating(s) = operating(s↼) — {reac}
        shut_down(s) = shut_down (s↼) ∪ {reac}
```

Figure 4-Specification of the SHUT_DOWN command

In the case of SHUT_DOWN it is only defined if *reac* is found in the set of operating reactors. The post-condition states what happens after an operation is executed. It uses the convention of placing a harpoon over the stored data. When a variable has a harpoon above it in a post-condition, it is a reference to the value of the variable *before* the operation was executed. Thus, the first part of the post- condition can be read as: 'after the operation has been completed, the set of operating reactors in the system is equal to the old value of the set of operating reactors minus the reactor that is to be removed.' Similarly, the second part can be read as: 'after the operation has been completed, the set of reactors which have been shut down is equal to the old value of the set of reactors that have been shut down to which *reac* has been added.'

The pre-condition and post-condition are part of the contract between the staff who specified the system and staff who are going to design and program it. It says that if the pre- condition holds, then the system should guarantee that the post- condition holds.

The specification for the START_UP command is similar – see Figure 5. In this case, the pre-condition states that the reactor reac must be shut down and there must be less than 20 reactors which are currently operating. The effect of the command, as detailed in the post-condition, is to remove reac from the shut_down set and add it to the operating set. The operation requires write access, since the data is modified by the operation.

The final operation OPERATING is slightly different to SHUT_DOWN and START_UP since it returns an integer value. This operation is shown in Figure 6. The first line includes the expression *no_in_operating:* $\mathbb{N}$. This signifies that the operation will deliver a value *no_in_operating*, which will be a natural number (an integer in the range 0 to ∞). The symbol $\mathbb{N}$ stands for the set of natural numbers.

Since OPERATING only queries how many elements there are in the set of operating reactors, the stored data is specified as read only. There is no pre-condition because the operation will work correctly irrespective of the state of the stored data. The post-condition states that after the operation has terminated, the variable *no_in_operating* will contain the number of elements in the operating set, ie the number of reactors that are operating.

Checking the spec

With the specification completed, the next step is to check that it is correct. The first check that is made is to ensure that the operations do not violate the data invariant

In a banking application, the data invariant might state that no account is allowed beyond a particular overdraft limit

– that they do not give rise to data for which the data invariant is not true. This is easy to show in the case of the OPERATING operation: it only has read access to the stored data, so the data must remain unchanged.

We can quite easily demonstrate informally that the data invariant holds for the START_UP command. First, we assume that it holds before the operation, and then examine what the operation does to the stored data. If the data invariant holds before the operation we know that no more than 100 reactors will be in the system. Now, since START_UP does not create any new reactors, after the operation has been executed there will still be no more than 100 reactors in the system.

Let us next examine that part of the data invariant which declares that *no more* than 20 reactors will be operating simultaneously. As before, assume that the invariant holds before the operation. Now, the pre-condition of the START_UP operation tells us that for the operation to be defined there must be *less* than 20 reactors in the set of working reactors. The post-condition states that a reactor is removed from the *shut_down* set and placed in the *operating* set; that is, that a single reactor is added to the *operating* set. Thus the number of reactors in the operating set *after* the operation has been completed must be *less than or equal to* 20. Hence this second part of the data invariant will hold after the operation.

The remaining component states that a reactor cannot simultaneously be operating and shut down (ie it cannot be both a member of the *shut_down* set and the *operating* set). Again, assume that the data invariant holds before the operation. After the operation has been executed, all that has happened is that a reactor has been transferred from the shut_down set to the operating set. Hence, if there are no common members of these sets before the operation, there must be no common members after the operation; so the invariant holds. A similar set of arguments can demonstrate that SHUT_DOWN will not violate the data invariant.

A second set of checks try to demonstrate that the properties of the system are as expected. In our example, one of these would be to prove that, if a reactor was shut down and then started up again, the stored data would not change from its initial state. Another possible check would involve demonstrating that, when a reactor was shut using the SHUT_DOWN operation, there would be one more member of the *shut_down* set than there was before the operation was executed. I will not show the reasoning required to demonstrate that these properties hold. Note that it could be done by the same informal process that used to demonstrate that the data invariant was not violated.

The example in this article is quite small. Informal reasoning is sufficient to prove that the specification is correct. As systems get larger, this type of reasoning must be augmented (and eventually supplanted) by mathematical proof. When it comes to very large systems, this proof becomes so diffi-

```
START_UP (reac: reactors)
ext wr s: system
pre     reac ∈ shut_down(s) ∧
        card operating(s) <20
post    operating(s) = operating(s↼) ∪ {reac} ∧
        shut_down(s) = shut_down (s↼) — {reac}
```

Figure 5-Specification of the START_UP command

```
OPERATING(reac:reactors) no_in_operating: N
ext rd s:system
post no_in_operating = card(operating(s↼))
```

Figure 6-Specification of the OPERATING command

cult that computer assistance is required. Some specialised software tools, known as theorem provers, have been developed to enable much of the proof process to be automated.

Once the specification has been checked, the process of specifying the functions of a system mathematically is complete. We can now produce a design which matches the specification. Development staff decide how the system is to be implemented. For example, they might decide that, because speed is of the essence, the reactor data is to be stored in an array, ordered by the name of the reactors, retrieved by a fast binary search.

Irrespective of the implementation, the data is again specified using mathematics, with design operations corresponding to START_UP, SHUT_DOWN and OPERATING described by pre-conditions and post-conditions. The checking process is then carried out, ensuring that the data invariants are not violated by the operations and that properties of the implementation are reflected in the design. At this stage another series of checks must be done: checks that prove that the design meets the specification. After this, the system can be given to a programmer to implement.

Conclusion

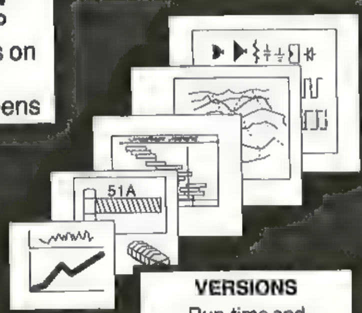
Of necessity, the example that I have given has been small. However, it has displayed all the features of formal software development: the use of mathematics, to specify the functions of a system and its design, and the use of proof techniques, to check that the descriptions produced at each stage in the development are correct and an adequate reflection of previous stages. EXE

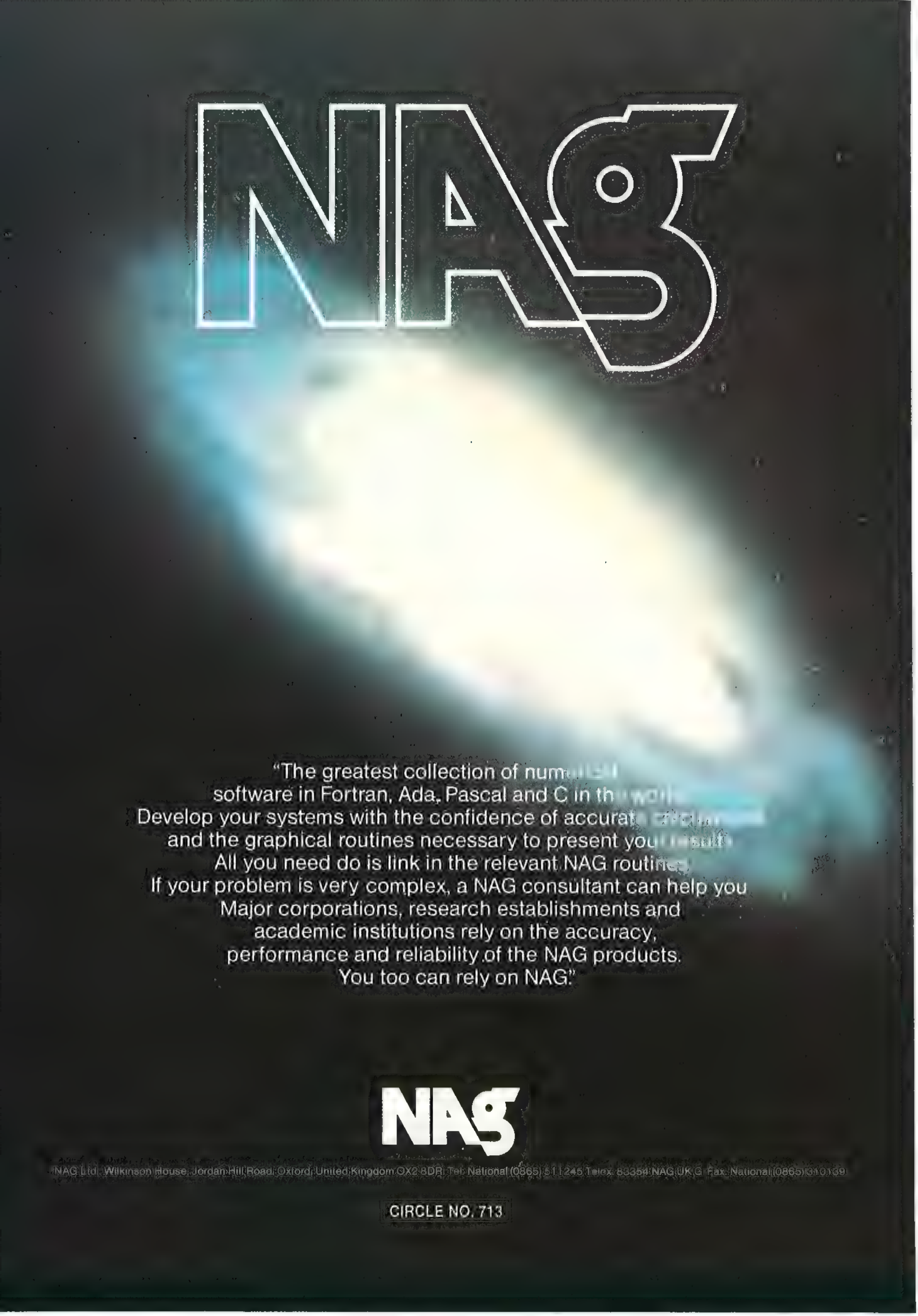
Darrel Ince is a Professor of Computing Science at the Open University, where he is the acting head of the computing department. If you are interested in further study of mathematical methods for software development, he recommends the three books listed below:

An Introduction to Discrete Mathematics and Formal System Specification. *(Darrel Ince, Oxford University Press). A book which teaches the mathematics behind formal methods and introduces the Z formal development method. Z is very similar to the notation presented in this article. The book is suitable for self-study.*

Systematic Software Development Using VDM. *(C Jones, Prentice-Hall). The bible and reference book for VDM. However, it does assume a good knowledge of discrete mathematics. If you're going to carry out serious software development using VDM then this book should be on your shelves.*

The Science of Programming. *(D Gries, Springer-Verlag). An excellent book, which describes a formal method of software development suitable for developing subroutines.*




NAG
"The greatest collection of num
software in Fortran, Ada, Pascal and C in th
Develop your systems with the confidence of accurat
and the graphical routines necessary to present you
All you need do is link in the relevant NAG routi
If your problem is very complex, a NAG consultant can help you
Major corporations, research establishments and
academic institutions rely on the accuracy,
performance and reliability of the NAG products.
You too can rely on NAG."
NAG
NAG Ltd. Wilkinson House, Jordan Hill Road, Oxford, United Kingdom OX2 8DR. Tel: National (0865) 511245 Telex: 83354 NAG UK G Fax: National (0865) 310139
CIRCLE NO. 713

dBASE and Clipper Tools

Bob Rimmington compares two productivity tools for dBASE developers that might be of use to you – providing you can work things out for yourself.

Development time in dBASE is costly, and a package that can save even a few days' work will soon pay for itself. Software houses have been quick to recognise a potential market for any language in wide use, and two quite different approaches are considered below. The first aims to write much of the more tedious program code for you, while the second tries to ease the headaches of designing reports.

UI Programmer

UI is an application generator. The user draws menus and data entry forms on the screen, and the software then writes the code. That is the essence; in practice there is a wide range of facilities and sophistication.

The package comprises a thick A5 ring manual and four disks. Installation creates a directory structure comprising: `\UI`, the main editing/help software plus utilities for documentation and conversion from UI 1.0, `\UI\TEM`, template files, `\UI\TLB`, template library files, `\UI\WW`, user-designed applications in internal format and `\UI\DBF`, example data files and data dictionaries.

Screen Design Editor

Neither the manual nor the disks contain any kind of tutorial. Instead, it is suggested that a new user loads one of the supplied example applications in the \WW directory, looks at it, modifies it and generates code from it.

Starting up UI reveals a blank screen except for cursor position and status indicators at the top right. This is, in fact, the design editor. All the tools and facilities are accessed either from a menu or, with more familiarity, with various key combinations. Pressing any function key or Alt-Z will display the menu as a bar across the top of the screen. The ten main options select further drop-down menus.

F1 accesses help, F2 loads and saves files, F9 generates code and F10 allows various configuration parameters to be set. The remaining keys are for the box, menu, data and similar options – see Figure 1. If, from the F6 'Box' menu, a 'New Box' is selected then further menus appear for border style, interior colour and box border. A box can then be stretched downwards and to the right from the current cursor position. One quickly learns that the cursor must be correctly positioned before selecting a tool as there is no way of repositioning afterwards. For boxes and some other objects this is not really a problem as tools to move, copy or re-size enable any required adjustment, but it is one of the many disconcerting features that are only discovered by trial and the all-too-often error.

The language is wide in scope and is explained at length with tutorials and examples

Designing in Practice

For the main editing screen, the manual and the help panels are for reference only with no 'getting started' guide. Perhaps it should be self-evident that an entry screen cannot be designed before creating a data file, but this and other less obvious prior requirements should be explained. For example, a menu box is not a box containing a list of options. It is, in fact, one option for which colours and other parameters must be individually specified. The action it will initiate should also have been decided prior to menu design. A completed option list can then be boxed if required. To discover how one menu can call another it is necessary to load and compile one of the example applications; it is not mentioned at all in the manual.

With practice, it is clear that most visual aspects of a program can be designed on screen. This could prove very useful when discussing a new system with a user or client, as an acceptable user interface could be designed and agreed as part of the specification prior to any conventional coding. Such layouts can include nested menus and multi-screen entry forms. Source code can then be generated in all the main dBASE flavours and, to an extent, in C. Initially, one of the supplied templates can be selected, later you can modify them or add your own. The code produced can go far beyond that needed for the screen itself. For an entry form, there are templates specifically for just Append or Edit, and also more comprehensive ones such as 'Basic'. This writes a self-contained sub-system with routines to establish memory variables matching all the field variables and a bottom line menu with options like Next, Previous, Last, Goto, Edit, Append and Delete. It may not suit all requirements, but is a quick way to produce a simple system or the basis of a more complex one.

Code Generation

The code itself varies according to the target software selected. It is commented,

Figure 1 – Menu facilities for UI programmers

F3 Edit	F4 Vars	F5 Data
Move area	Add field	Attached data dictionary: <none>
Copy area	Add bunch of fields	Modify attached data dictionary
Erase area	Define new memvar	Attach existing dictionary to form
Hide objects	Revise definition	Attach new (empty) dictionary to form
Show objects	Move	Local data definitions
Remove objects	Copy	
Depth-O-Scope!	Erase	
Paint-O-Matic!	Show Pictures/names	
Paint area	Set say/get order	
B'ground color	Display color	
Zap		

F6 Boxes	F7 Menus	F8 Other
New border box	Create new option box	DOS vacation
New no border box	Attach option definition	Draw single line graphics
Adjust shape	Revise definition	Draw double line graphics
Move	Delete definition	Draw blanks
Copy	Unselected option color	Draw any character
Erase	Menumatic!	Turn line drawing off <esc>
Unbox	Sequence options	Join lines at cursor
Fill from below		Join everything
Outline color		Center line under cursor
Inside color		Zoom in on box or var
Revise definition		Turn info line on/off
Tag/untag as popup		License plate
Hide		
Hide pile		

and broken into procedures where appropriate. It may well prove far lengthier and slower than your own. Each menu or screen is fully coded as if it was a one-off, with much consequential duplication. A routine such as `REPLACE <field> WITH <memvar>` puts each `REPLACE` on a separate line rather than place them in one command statement. The `<memvar>` equivalents to fields comprise the field name with an 'm' prefix so that the last digit of a 10 character field name ceases to be significant. `'STOCKWEEK1'`, `'STOCKWEEK2'` ... `'STOCKWEEK9'` all effectively become `'mSTOCKWEEK'`. This sort of error is not reported and there is no warning in the manual.

For dBASE users and other clones without the Clipper `MENU TO` and `PROMPT` commands, the ability to produce bounce-bar and similar menus so easily will be welcomed. A Data Dictionary feature is only partly explained, but can be used to add `PICTURE` and `VALID` to `GET` statements and to specify indexes and relationships for the corresponding .DBF file.

The Template Language

To modify or write templates, a combination of normal dBASE code and the UI template language is used. The latter is identified by placing it within `<<UICODE>>` and `<<ENDUICODE>>` statements or it can be embedded in dBASE code by placing it inside `<< >>` symbols. During generation of source code, any normal dBASE code is written straight to the output file, but template code is interpreted using .TLB (Template Library) files. For example, `<<display-box(bar)>>` calls the Template Library function 'display-box' to expand the menu, as drawn on the screen and named 'bar', into conventional dBASE code. The language is wide in scope and is explained at length with tutorials and examples. Indeed 80% of the manual is devoted to template explanation and reference.

The Verdict

UI is a powerful product that will appeal to many, but not all. As a productivity tool it is not so easy and intuitive to learn as is implied,and it tends to produce a stream of flabby code. If you write each application from scratch it may well save time, but not if you largely draw from and modify existing routines. Indeed, programmers who already have their own library of routines and methods may not find it easy to make use of UI beyond the screen design level. Others will find it useful for quick development or for users, with a little help, to design simpler systems for themselves. A development group may well find it worth programming their own templates to give a 'house style' to each team member's routines.

R&R Report Writer

The name describes the product well. It enables reports to be designed without the need for conventional programming code. The basic system comes on three disks with a substantial manual. To use it with Clipper or FoxBase index files, an additional module is required. This comprises a disk with two 1K files and a single instruction sheet which is not a lot for an extra £50. An optional source code generator comes on a further disk with a 70 page booklet.

Installation is straightforward with clear instructions. Once completed, typing 'RR' takes you straight into the main edit screen. At the top is a three line control panel, on the left a line type column – see Figure 2. The remaining space is the work area. It is used rather like a spreadsheet without any cells. Text such as headings can be typed straight into the work area, while most other operations are accessed by pressing the '/' key and selecting from menus displayed in the top control panel. Anyone familiar with Lotus 1-2-3 will feel at home straight away.

Getting Started

Line types comprise Title, Header, Body, Footer and Summary, and are selected through the 'Line – Create' and then 'Line – Insert' menus.For a simple columnar report fields can be placed on a 'Body' line by positioning the cursor, pressing the F10 (Options) key and selecting a field from a pop-up list for the previously specified .DBF file. If it does not look right, '/ Move' will re-position as required. Further body lines can be used if needed, eg for addresses. Next, add some header lines and type in a title and some column headings. Add some footer lines if required. Then '/ Print – Go' will scroll your report up the screen so that you can see just how it will look. It is as easy as that.

Complex Reports

It is with more complex reports that the full power of R&R is revealed. The scope for setting relationships is particularly good. A Master .DBF file is defined. Other data files may be linked to it, and they in turn can be linked to one another. The relationships can be on a common or calculated field, they can also be one of three types. Exact is self-evident, Approximate can be used with a quantity range to find, say, a discount level; Scan will search for all matching records in the linked file. For all types, three Failure actions are available. Blank (omits data from the linked file),

Skip (moves to next record in Master file) and Terminate (prompts user for action). A nice touch is that, as criteria and rules are selected, a paragraph describing them in normal English is shown at the foot of the screen. Internally R&R builds a composite record from the linked files which is displayed in the pick-list used for placing fields in the report.

Also included in the pick-list are calculated fields. These can be typed in (after selecting '/ Fields – Calculate – Create'), but it is easier to use the F10 (Options) facility. Through these fields, operators, functions and expressions (such as a previously defined calculated field) can be picked to build the required new expression. About 75 built-in functions are available. Some are familiar ones such as `VAL()` and `TRIM()`, others are unique to R&R and include `ADDDAYS()`, `DAYSBTWN()`, `INLIST`, `INRANGE`, `CASE()`. Clearly the scope is very wide. Calculated Total fields can also be defined from options such as Sum, Count, Average, Variance. There can be a grand total for a whole report, reset for each page or reset for up to eight different group levels.

Queries and Groups

The query menu enables a subset of the data to be defined. First a field is selected from a pick-list and then a 'comparison type' list is displayed. Besides the expected 'equal to' and similar, there are in/not-in 'Range' and 'List' options. The latter will accept a list of alternative criteria. For example, to obtain a random 10% sample, a survey could specify selection by birth dates of 5th, 15th or 25th of any month. Not an easy task for many ad-hoc query tools, but no problem for R&R. A Date-of-Birth field comparison with a list of 05/*/*, 15/*/* and 25/*/* (note the * wild card) extracts the required records.

Although R&R will expect normal .NDX or .NTX index files to exist when relationships are defined, it provides its own sort facilities for reports with temporary index files. Selection through the normal menu and pick-list facilities allow a sort hierarchy of up to eight fields. Grouping criteria defaults to the sort fields but can be changed, eg Sort by date and Group by month. Separate headers and/or footers can be defined for each of up to eight levels of groups, and can include data in the form of previously defined 'Calculated Total' fields. Page totals and a final Summary can also be included.

Others will find it useful for quick development or for users, with a little help, to design the simpler systems for themselves

Documentation

The 400 page manual is straightforward, clear and comprehensive. It appears to cover every aspect, has examples where needed and a proper index. It includes a tutorial chapter for use with supplied report and data files. Learning from printed instructions may seem rather old fashioned but do not be put off, it is very good. As one follows the clear instructions for creating the example reports, there are reminders to look at the Help screens for further guidance. These too are good, are not over verbose, but include a reference to the manual pages containing fuller details. Even when designing the relatively complicated multi-line invoice report, the screen layout and cursor position always exactly matched the illustrations in the tutorial.

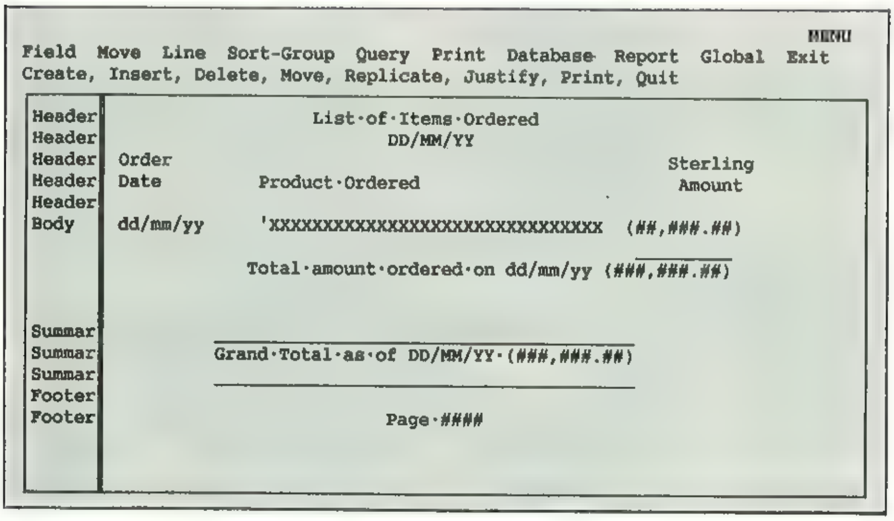


Figure 2 – RR Report Writer in use

Options

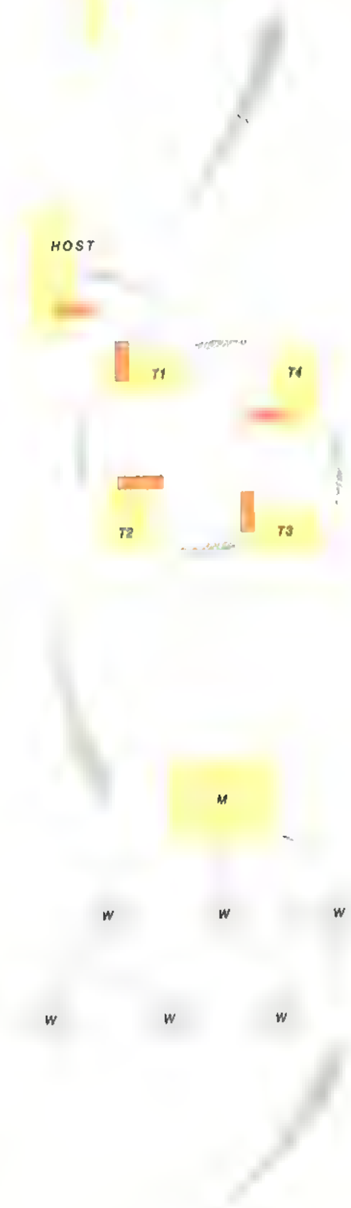
During Setup, although screen, index types and other details can be specified, most questions relate to printers. Standard configurations for up to eight different printers may be selected from a supplied list of around 60. This sounds a lot but many commonly used printers are missing, eg the HP Deskjet and the current Star range. It is possible to supplement these configurations or write your own – this seemed necessary in all cases for enlarged and NLQ styles. Pitch is limited to 10, 12 or compressed and applies to the whole document. Lines may be only six or eight to the inch and cannot be specified in n/72 or n/216 units. This will be a problem when a report must fit a pre-printed form.

During design of a report, more print options are accessed through menus. It is possible to set currency to the '£' symbol and date format to the British style. There is also provision in the 'Field' and 'Line' menu structures for attributes and fonts to be applied to specified fields/text and lines. In all respects except one, the range of options gives great flexibility and will satisfy most requirements. However, if you need to chop and change the number of characters or lines per inch within a report, you are probably back to conventional coding.

Performance

R&R proved a robust package and tolerant of initial user errors. It is relatively slow at processing its own internal sort and query operations, and this might prove a problem with large files. Its otherwise impeccable tutorial revealed an internal logic error. In the header of an invoice the `ADD-DAY()` function correctly added the specified 30 days to the order date. When used again at the foot, it showed a different date, apparently that of the next record as, by then, the file pointer had moved on. The limitations on formatting with font changes are a disappointment. I find that getting this right is often the most time consuming part of print routines and it should be possible to provide a tool to help with this.

THE TRANSPUTER COMPILERS



TRADITIONAL LANGUAGES

Many applications developers world-wide are using the abundant computing power offered by multiple transputer systems. It is sad that many others reject this technology because of the mistaken belief that you can only program transputers efficiently using the occam language. In reality you can write efficient transputer applications entirely in the traditional languages C, Fortran or Pascal using 3L parallel compilers.

EXISTING SKILLS

Transputer applications are built from conventional sequential programs sending data to each other. Such parallel programming needs sequential techniques with which every programmer is familiar. You can use existing programming skills and often even existing code.

FAMILIAR ENVIRONMENTS

Transputers are usually accessed from host computers running standard operating systems. The most common hosts are PCs with MS-DOS but OEM versions exist for others including SUN, Apollo and VAX. 3L compilers and applications developed with them appear as host commands. This allows you to work in familiar environments using host software such as editors and file management utilities.

DEVELOPMENT ASSISTANCE

To assist program development 3L provides Tbug, a powerful source-level debugger. Tbug is an interactive, window-based tool which helps you to observe and control the execution of parallel programs containing many components running concurrently.

FLEXIBLE CONFIGURATION

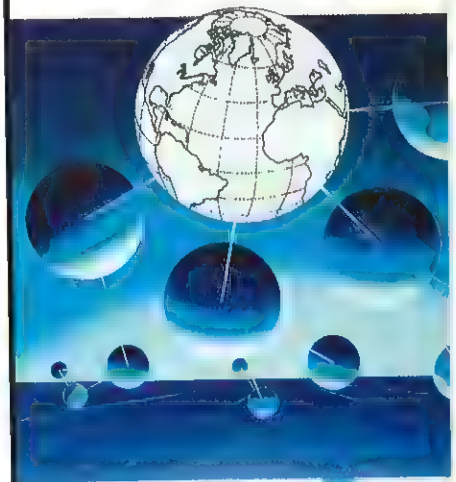
3L's configuration tools let you match applications to existing transputer networks or use processor farm techniques where applications automatically use every available transputer. Processor farms can also be used to run multiple copies of a program in parallel, rather than reinvoking a single instance many times

WORLD LEADER

The popularity of 3L's transputer development tools, either bearing the 3L name or repackaged under licence, has made them world leaders. 3L software is available from over 70 distributors world-wide.

COMPLETE PACKAGES

3L's compilers and configurers come as complete packages; no extra system software is required to write applications making use of multiple transputer networks. Tbug is available separately. All 3L software is compatible with the majority of transputer add-in boards commercially available.



3L LIMITED
PEEL HOUSE
LADYWELL
LIVINGSTON
WEST LOTHIAN
EH54 6AG
SCOTLAND
TEL 0506 415959
FAX 0506 415944

CIRCLE NO. 717

Three Ways of Using R&R

R&R can be used entirely as a stand-alone product. It needs data in the form of .DBF, DBT and index files, but does not require dBASE or other software. This is the mode in which reports are designed; it can also be used for all printing. Indeed there could be circumstances where, for a user who requires only output from data on supplied files, R&R might be a better solution than a database package. Again, an application written in Clipper might handle all data entry and regular reports but not cope so well with ad-hoc needs. Provision of R&R could be a much better solution for a moderately competent user than trying to include a complex query module within the application.

The standard R&R package includes run-time files that may be distributed to users with pre-designed report files. It is possible to print or display straight from DOS but most will wish to do so from within a database application. This is done through use of the 'RUN' command, with parameters, and requires at least 25K of free RAM. A range of options provide user control and error trapping for flexibility. No problems were encountered using this simple and economical method.

For dBASE IV (not III+), Clipper, FoxBase, Quicksilver and dBXL, a separate code generator is available. As with R&R itself, it is easy to install and use and, as a means of obtaining code for inclusion in a compiled application, it does look attractive. However, its use as an alternative to the run-time method is debateable. The code it produces is lengthy and, where several reports are involved, repetitive. There is a facility to omit and generate separately any common functions, but this only a partial solution. To avoid excessive growth in the size of the compiled .EXE file, a fair amount of manual editing would be required, never too easy with unfamiliar code. Even then, for more than one or two reports, it may prove a less efficient use of RAM than the run-time alternative. There is also less error trapping – one test failed through a numeric overflow that, in run-time only, gave a nonsense figure.

Conclusions

R&R is undoubtedly one of the more useful dBASE add-ons and its suitability as a user tool as well was an unexpected bonus. As with many such packages, if you have already perfected your own library of print routines you may not consider it worth changing. However, if you find that producing all those reports the typical user demands is a time-consuming chore, then R&R is probably the best solution around.

Both UI and R&R are powerful tools. The main 'pro' is that, for many programmers, they should both ease and speed development of a new system. The 'cons' are less easily defined. The code they produce may not fit comfortably within existing and familiar programming methods. There will be unusual or complex situations where they will not cope, so leaving the programmer to tackle the difficult before he has cut his teeth on the commonplace. A decision on purchase will, therefore, depend more on how you view the pros and cons than the products themselves as most will find these satisfactory.

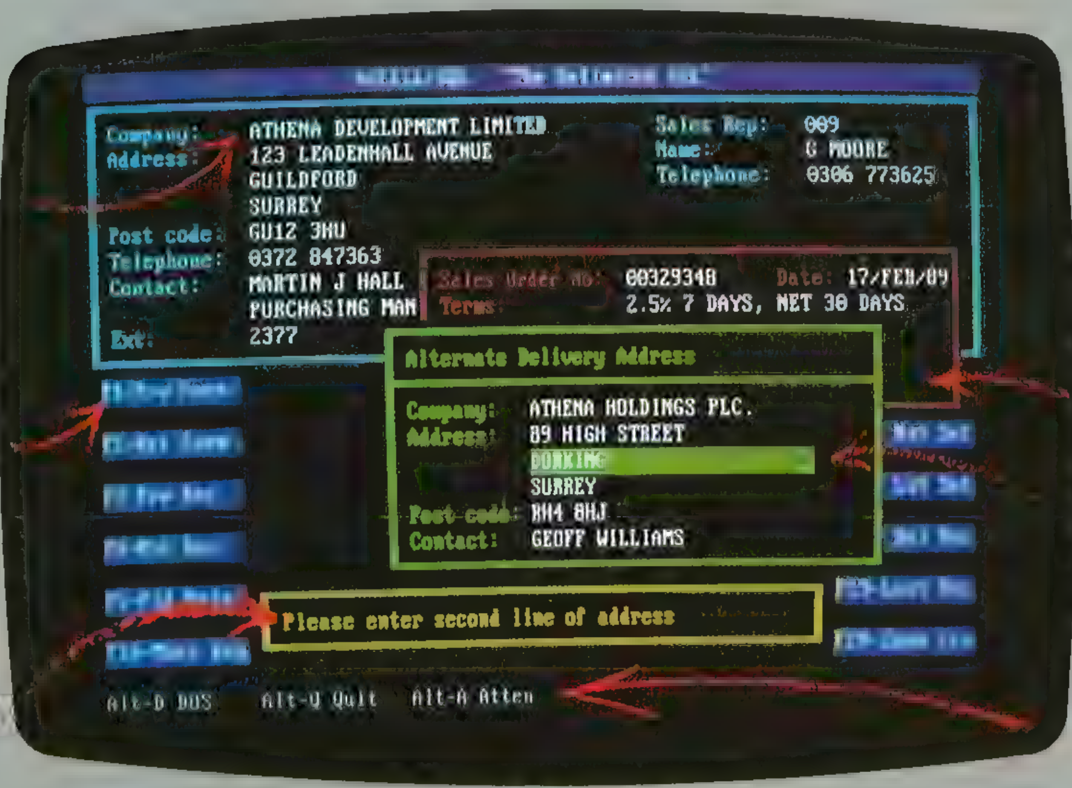
Bob Rimmington is an independent consultant. He can be contacted through Stanford Systems on 0444 236352. *Both the UI Programmer and R&R Report Writer may be obtained from Software Paradise, who are on 0222 887521 (Fax 0222 862209). UI Programmer costs £295, and R&R Report Writer is £135, plus £49.95 for disks required to run Clipper or FoxBase (Full package £179).*

"WHAT IF YOU COULD DEVELOP AN APPLICATION THAT OPERATES AT THE SPEED OF SIGHT?"

Query By Forms interactively finds or updates data with just a few keystrokes

Fully configurable function keys and status messages

Field-sensitive Help windows enable novice users to make veteran decisions



Exclusive Zoomview feature lets users scan and retrieve data from look-up windows without leaving the original transaction.

Full overlapping window interface provides PC-type flexibility.

Horizontal and vertical scrolling areas for optimum screen space usage.

Co-operative Processing ensures maximum host throughput in network environments.

YOU CAN WITH ACCELL/SQL

The New Integrated Development System

ACCELL/SQL fully integrates a powerful application generator and a flexible 4GL.

Providing a unique UNIX-based development environment for portable applications which will perform across a range of leading SQL Relational Databases with a variety of popular interfaces including Microsoft Windows.

UNIFY's own RDBMS is the new UNIFY 2000 database engine-designed to support large, complex systems requiring 24 hour availability.

If you want to create powerful, user-friendly applications while greatly reducing development and maintenance time, get in touch with UNIFY.

UNIFY Corporation
Prestige House 23-26 High Street
Egham Surrey TW20 9DU
Telephone: 0784 71021
(International 44 784 71021)
Fax: 0784 437590

UNIFY CORPORATION

UNIFY Corporation is a leading manufacturer of independent UNIX-based application development software tools.

CIRCLE NO. 719

Trap Those Caps

If you're troubled by a Caps Lock key that's all too easy to hit without looking, Robert Schifreen has a solution.

How often do you look up at the screen, only to find that you've inadvertently hit the Caps Lock key and the last two lines of source code that you typed are all in capitals? Never? If so, then you don't need the following program, so go straight to STOB and don't collect £200.

Assuming you do need CLBEEP, type it in and assemble it to a .COM file. Install the program by running it from the DOS prompt. From now on, you'll hear two beeps whenever you hit the Caps Lock key. The two beeps will rise in frequency when you turn Caps Lock on, while the second will be lower than the first when turning it off.

Caps Lock is normally pressed from within an application such as an editor, rather than when you're at the DOS prompt. Therefore, it's not safe (and it's quite boring, too) to use a DOS call to print a Ctrl-G in order to beep. The sound routine used here uses direct hardware access only, so it's safer and more flexible. Even so, it may not be a good idea to keep your finger on Caps Lock while saving a large file to a hard disk. I've been using the program daily for a month, as have half a dozen colleagues, and no problems have been detected.

The program works by trapping interrupt 9h, which is the one that's generated by the PC whenever you hit a key. Note that no program ever actually issues an INT 9 command – the handler is called automatically when a key is pressed. The Int 9 handler's sole job is to retrieve the key from the keyboard port and place it in the DOS keyboard buffer, ready for an Int 16h call, issued by an application, to retrieve when it wants one. CLBEEP watches for Caps Lock being hit, and calls the beep routine before chaining on to the original handler, which will process the key press and place it in the buffer if required.

HISTORY Problems

There's a problem with the HISTORY program published in Code Page in October's issue. The `/Q` switch, which suppresses the sign-on text, doesn't have any effect if you run HISTORY from a batch file. To ensure that the sign-on text is suppressed, use the `HISTORY >NUL` command instead if you run the program from your AUTOEXEC.BAT file and don't want the messages displayed.

```
; CLBEEP.ASM - BEEP WHEN CAPS
; LOCK HIT Version 1.0, November
; 1989. Assemble to a .COM file.

 code   segment
assume  cs:code,ds:code
org     0100h
start:  jmp init
intent:
; Come here when Int 9 generated

        pushf
        push ds
        push cs
        pop  ds
        push ax
        push bp
        in   al,60h
; Get scan code from keyboard
; hardware
        cmp  al,0BAh
; scan code of key being detected,
; plus 80h 'cos we're detecting
; release code not the press code.
        jne  kb_exit
        call sound
kb_exit:
        pop    bp
        pop    ax
        pop    ds
        popf
        db     0EAh
; Far jump to original Int 9
; handler
intip   dw     0
intcs   dw     0
; Address of orig handler

blip proc

; Code to make a bleep.  Remember
; that this procedure could be called
; from anywhere, so we have to use
; direct hardware access and avoid
; all DOS and BIOS calls.  Frequency
; is expected in di (in Hz).
        mov    bx,30
; duration in 1/100 secs
        mov    al,0b6h
        out    43h,al
        mov    dx,14h
        mov    ax,4f38h
        div    di
        out    42h,al
        mov    al,ah
        out    42h,al
        in     al,61h
        mov    ah,al
        or     al,3
        out    61h,al
 blip1:
        mov    cx,2801
 spkr_on:
        loop   spkr_on
        dec    bx
        jnz    blip1
        mov    al,ah
        out    61h,al
        mov    cx,55000
fred2:
        loop   fred2
; delay a while after beeping
        ret
blip    endp
        sound proc
; Called when Caps Lock pressed.
        push  ax
        push  bx
        push  cx
        push  dx
        push  di
        push  es
        mov   ax,40h
        mov   es,ax
        mov   al,byte ptr es:[17h]
; get shift status byte
        and   al,01000000b
; isolate caps lock status bit
        cmp   al,0
; Is caps lock already on?
        je    is_on

        mov   di,800
; Make a low beep (800 Hz)
        call  blip
        mov   di,2000
; And a high one
        call  blip
        jmp   short fred
; We're done
is_on:
        mov   di,2000
; Make a high beep
        call  blip
        mov   di,800
; And a low one
        call  blip
 fred:
        pop   es
        pop   di
        pop   dx
        pop   cx
        pop   bx
        pop   ax
        ret
        sound endp
init:
; Install CLBEEP in memory
        mov   ax,cs
        mov   ds,ax
        mov   ah,35h
;Get old interrupt vector
        mov   al,9h
        int   21h
        mov   word ptr cs:intcs,es
; Save in long JMP
        mov   word ptr cs:intip,bx
        mov   ah,25h
; Set new interrupt vector
        mov   al,9h
        mov   dx,offset cs:intent
        int   21h
        mov   ax,3100h
; End, with errorlevel 0
        mov   dx,(init-start)/16+17
; Number of resident paragraphs
        int   21h
        code  ends
        end   start
```

CLBEEP.ASM

Here are the bugs: Turbo C 2.0

This month, our bug-hunting department concentrates on Borland's Turbo C. Neil Martin is the man with the magnifying glass.

In October's issue of .EXE, I demonstrated some of the more dangerous bugs in Microsoft C 5.1. This month I shall be digging up the dirt on Borland's popular Turbo C compiler. Turbo C started life as Wizard C, which was a successful command line-based compiler in its own right. Borland took over Wizard Systems Software, and the compiler was renamed Turbo C. You only have to look at the dates displayed by the two compilers to see what happened:

```
CC Version 3.0b
Copyright (c) 1984, 1985, 1986
Wizard Systems Software, Inc.
```

```
TCC
Turbo C Version 2.01 Copyright
© 1987, 1988 Borland International
```

```
#include<stdio.h>

main()
{
  static  long array[] =
    { 0xf0010000, 0xf0020000,
      0xf0030000,0xf0040000};
  int     i ;

  for(i =0 ; i<=3; i++)
  {
    switch( array[i] )
    {
      case  0xf0010000:
        puts("case 1");
        break;
      case  0xf0020000:
        puts("case 2");
        break;
      case  0xf0030000:
        puts("case 3");
        break;
      case  0xf0040000:
        puts("case 4");
        break;
      default:
        puts("We shouldn't be
              here");
    }
  }
} /* main */
```

Figure 1 – The Switch Statement

The most important feature which Borland added to its newly-acquired Wizard compiler was the integrated environment. This was a major leap forward in the evolution

Turbo C gives a redeclaration message when enumerated types are used in prototype scope

of user interfaces and, with the exception of JPI's TopSpeed C (not yet released), I do not think any other MS-DOS based C compiler has come close to it.

The version of Turbo C that I used to prepare this article is an interim release known as 2.01. You may be interested to know that the very existence of this release is due, in part, to a letter published in .EXE (April 1989). This drew attention to a serious bug in version 2.00 of the compiler and took Borland to task because their technical support department could not help. The bug has been fixed in 2.01, and the good news is that Borland has said that it will upgrade registered owners of 2.00. If you think that you are eligible, please contact Borland.

In case you missed the original letter, the problem was this:

```
extern int g [80] ;
int a, b;
b += g [a] ;
```

When the above code was compiled using the huge memory model, the assignment did not work. I can confirm that this works properly with 2.01, though. Now for some things that do not.

Bug #1: Big Switch

The Bug. The `switch` statement accepts long integers as described in the draft ANSI standard. Unfortunately, the code that is generated does not always work. The program in Figure 1 demonstrates the problem. The output should, of course, be:

```
case 1
case 2
case 3
case 4
```

Turbo C actually says:

```
case 4
case 4
case 4
case 4
```

The Fix. Stick to pre-ANSI limits for `case` statements, ie values within the range `INT_MIN` to `INT_MAX`.

Bug #2: Out of sizeof

The Bug. The `sizeof` operator does not always return the correct size. The code in Figure 2 demonstrates the problem. This outputs the following:

```
sizeof float = 4
sizeof char = 1
sizeof +char = 1
sizeof -char = 2
sizeof char+float = 8
```

The last two results must be described as `weird` and `weird++`.

Figure 2 – The Sizeof Operator

```
#include<stdio.h>
#include<stddef.h>

main()
{
  char c;
  float f;

  printf("sizeof float = %d\n", sizeof(f));
  printf("sizeof char  = %d\n", sizeof(c));
  printf("sizeof +char  = %d\n", sizeof(+c));
  printf("sizeof -char  = %d\n", sizeof(-c));
  printf("sizeof char+float = %d\n", sizeof(c+f));
} /* main */
```

Figure 3 – Initialising const strings

```
#include<stdio.h>

main()
{
           static char ss[] = "a static string";
  volatile static char vs[] = "a volatile string";
  const    static char cs[] = "a const string";

  puts(ss);
  puts(vs);
  puts(cs);
} /* main */
```

Figure 4 – The Remainder Operator

```
#include<stdio.h>

main()
{
  auto unsigned short us = 0x80e6;
  auto short s = 13;
  static short * ps;
  ps = &s;    /* *ps now is 13 */

  printf("%d\n", (13 %13));
  printf("result 1 = %d\n", ( us <= (s % *ps) ) ^ 0);
  printf("result 2 = %d \n", ( us <= (0) ) ^ 0);
} /* main */
```

The Fix. If you are reading or writing file data, this bug could prove to be quite a problem. The best suggestion I have is to treat the `sizeof` operator with extreme caution.

Bug #3: Const strings

The Bug. When the compiler option **merge duplicate strings** is used (-d switch with TCC command line version), the compiler fails to initialise `const` strings. For example, look at the code in Figure 3. Without the **merge duplicate strings** option, this produces the following output:

```
a static string
a volatile string
a const string
```

With the option switched on, however, the output is:

```
a static string
a volatile string
```

The Fix. Simple. Do not use the **merge duplicate strings** option.

Bug #4: Prototype scope

The Bug. Turbo C erroneously gives a redeclaration message when enumerated types are used in prototype scope. Consider the following code:

```
#include <stdio.h>

enum {a,b};

int f(enum{a,b});

main()
{

/* some code ....... */

}
```

When you try to compile this, you get `Redeclaration of 'a'` and `Redeclaration of 'b'` error messages at the line of the `f()` prototype.

The Fix. You cannot use `enums` in prototypes.

Bug #5: No Remainder

The Bug. The remainder operator `%` does not work properly in complex calculations. Take a look at the code in Figure 4. In the example, the expression `(s %*ps)` is actually `13 % 13`. Clearly, it should give a result of zero, and result 1 and result 2 should be the same. However, when you run it, the program prints:

```
0
result 1 = 1
result 2 = 0
```

The Fix. Rather vague, I fear. Avoid complex calculations that make use of the remainder operator.

Bug #6: Stupid enums

The Bug. The draft ANSI C standard states that the value of an enumeration constant should be representable as an `int`. The following should not, therefore, compile:

```
#include <stdio.h>
#include <limits.h>

main()
{
 enum{max= INT_MAX,
      max_1,
      max_2};

 if (max > max-1)
   puts("stupid enums");
}
```

Given that enumerations normally count upwards, the result is meaningless and most definitely non-ANSI.

The Fix. Enumerated types are, in general, not portable, even with ANSI C. My advice is to avoid them altogether.

EXE

Neil Martin is Senior Software Engineer with BSI Quality Assurance in Milton Keynes. He can be contacted on 0908 220908.

We need your help! If you have encountered a problem with a compiler (it need not be C) or programmer's tool (linkers, librarians, Windows API etc), .EXE would very much like to hear about it. Please send precise details of both the compiler and the bug, including a suitable code fragment, to the Editor at the address on page 2. We will publish such material in future articles, as well as forwarding relevant matter to the BSI.

A Software Robot

Don Milne looks at Robot, a PC package designed to 'use' your PC for you. Could this be a quick and reliable way to test your software?

Robot's main purpose is to make life easier for computing neophytes, by reducing the number of keystrokes required to perform repetitive tasks. However, Information Systems Ltd, the vendor,points out that automation of a PC can have many other uses (limited only by your imagination), and the role I will be looking at in particular is the potential for Robot as a tool to be used during the debugging and testing phase of software development. I suspect that few readers of .EXE Magazine consider themselves beginners at the computing game, so the usefulness of the product in this latter role is the one which is likely to be of most interest.

Most experienced DOS users are familiar with automation through the use of batch files which cut down on the keystrokes required to run any application. A batch file might change to a particular directory, rename or copy the odd file, install a mouse driver and then run the correct program. At that point, however, the automation is suspended until the application terminates, because DOS batch files control only the DOS command interpreter and not programs spawned from there.

The Robot command language is not so limited. Robot's command interpreter is a memory resident program which sits in the background, feeding keystrokes to a foreground application such as the DOS shell or any other application, whenever the foreground asks the BIOS for a key (Robot also has an optional feature which allows keystrokes to be fed to an application which traps Int 9h directly and reads the keyboard buffer). This is normally all that is required to control any package, provided it does not require a mouse.

If you have a simple repetitive task which you perform regularly, such as running a communications package to download all your messages from several bulletin boards, then Robot can very likely turn this into a single-keystroke operation. Robot scripts can be designed to run from the DOS command line, they can be picked from a simple menu, or they can execute whenever a particular hot key combination is detected. Scripts can also be designed to wait for a certain event to occur, such as a programmed timeout or a certain time of day, week or month, or when a particular message appears on the display. In style, the Robot language is reminiscent of the script language for a communications package, and in fact this is one of the suggested uses for it, to which the manual pays particular attention.

If a programmer knows that a bug is there he will examine the code or will get stuck in with a debugger

If that was all that Robot was used for, it would probably not merit coverage in a technical journal such as .EXE. However, because a Robot script can automate the operation of any program, it should be possible to construct test suites to exercise all the functions of any software that you develop. When you compile a new version of the software, you can have Robot run through all of the program features, making sure that the display appears exactly as it should, and that no new bugs have been introduced – Robot can do all of this unattended. I'll cover this in more depth later.

Installation

Unusually for these days of monster applications, Robot comes on a single 360 K disk, of which 218 K is unused. The first copy that I attempted to review was copy protected, using the kind of protection scheme which allows you to install on the hard disk just once. I duly installed the program on my hard disk, and everything seemed to work fine. However, when next day I tried to use the package, it refused to let me do anything – I could not run the program, uninstall or install it again. I called ISL and waited a couple of days for a replacement, only to have it do exactly the same thing again. It so happens that I make regular use of a disk defragmentation utility called SST, which is available in the public domain, and which I have found to be quite reliable on my equipment (you can download this program from the PC conference on CIX as well as other boards). I was reasonably certain that the Robot copy protection scheme, which I knew was creating a couple of hidden files, was making assumptions either that the hidden files would not be physically moved around on the disk, or that their directory entry would not be moved, or both. Neither assumption is in my opinion a particularly clever one to make, and it appeared that my use of a disk defragmenter and directory sorter (SST performs both functions) was the cause of the upset. I did not intend to stop using my precious tools because of a copy protection scheme.

Happily, when I made these points to a friendly person at ISL I was told that there had been other problems with the copy protection scheme and that they were about to drop it – so I asked for, and received, a disk which was not copy protected. To be fair to ISL, they are not the only people

```
-- source
WHEN TIMEOUT   0  0 2
  INJECT STRING "d:<CR>"
  INJECT STRING "cd \turbom2<CR>"
  INJECT STRING "ody startscr<CR>"
  PAUSE   0  0  2
  INJECT STRING "<@N>"
ENDWHEN
-- debug dump of object code for above script (89 bytes)
 57 00 00 02 09 00 47 58-00 49 08 64 00 3A 00 0D   W.....GX.I.d.:..
 00 00 00 49 1A 63 00 64-00 20 00 5C 00 74 00 75   ...I.c.d. .\.t.u
 00 72 00 62 00 6F 00 6D-00 32 00 0D 00 00 00 49   .r.b.o.m.2.....I
 1C 6F 00 64 00 79 00 20-00 73 00 74 00 61 00 72   .o.d.y. .s.t.a.r
 00 74 00 73 00 63 00 72-00 0D 00 00 00 70 00 00   .t.s.c.r.....p..
 02 49 04 00 31 00 00 45-45 00 00 00 00 00 00 00   .I..1..EE.......
 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

using this kind of copy protection, and they were very prompt in sending me replacement disks.

Provided you make sure that you receive the unprotected version, installation is very simple – just create a directory and copy all the files into it. Robot expects the home directory to be called C:\ROBOT, but you can override this default with an environment variable should you so wish. I was happy with the default, although I think it might have been smarter if this default were simply \ROBOT.

I really do hope that ISL are now shipping the unprotected version to customers. No programmer in possession of the majority of his or her marbles would allow themselves to become dependant on a piece of copy protected software. A programmer's machine is used to run untested software for the first time, when almost anything can happen (and does). The last thing such a programmer needs is to have his copy protected debugging aid wiped out by an errant program, with no way to recover the situation short of spending all night cracking the copy protection, or waiting several days for a replacement disk.

Programs

The Robot package consists of a suite of programs which can be operated on their own, or from a more friendly menu-driven front end. FRONT, is where most newcomers to Robot will probably start. This program is colourful and reasonably friendly, but the multiply nested menus will probably become quickly cumbersome as experience is gained. This is no problem, because the other programs in the suite can be invoked directly from the DOS command line, and scripts can be created using your favourite editor, provided it can produce plain ASCII files. FRONT is more than a menu system – it includes what amounts to a syntax directed editor, and it also allows you to automagically insert portions of display images and keyboard sequences created using the SNAPSHOT and REMEMBER utilities (see below).

The most commonly used programs in the suite will be COMPILE and ROBOT. COMPILE takes an ASCII source file and converts it to a tokenised form. As a brief aside, I would quibble with the use of the word 'compile' for this, there is nearly a one for one correspondence between source words and object tokens (notice how the word WHEN becomes the character 'W' and the words INJECT STRING become the token 'I'), so COMPILE no more compiles source files than the BASICA interpreter does when it loads and tokenises an ASCII format BASIC program.

I feel that the most promising possibility for Robot lies on the quality assurance side

When COMPILE has finished its task, it writes out the final object form of the script as a file with a .ROB extension – this is the file that ROBOT (the TSR) needs in order to execute your script. Robot takes as a command line argument the name of a .ROB file as created with COMPILE. Robot then becomes resident in memory and executes the named command file. In the case of the example shown in Figure 1, the script waits for two seconds and then changes directories and runs a program. This particular script is rather naive in that it issues commands which can only be understood by the DOS shell without first checking what the PC is doing at the time – this would cause a confusing mess if you happened (say) to be running your favourite editor at that moment, or even if you were in the middle of typing some other DOS command. However, it is possible to insert all the appropriate checks which prevent such things from happening.

There were two things I found disturbing about the Robot memory resident intepreter. First is that, contrary to my expectations, the TSR did not unload itself when the script completed. As a programmer I know that DOS has problems in this area (you can mess things up if a TSR releases memory when another application is in control of the machine), but I would have thought it possible for the interpreter to keep an eye open for an opportunity to unload itself, say when control returns to the DOS prompt. Nor could I find any mention in the manual of a way to unload the TSR, short of resetting the PC.

Even more disturbing was that multiple uses of the Robot command from the DOS prompt caused multiple incarnations of the TSR to become resident in memory. I would expect the new script to be added to a list of tasks for a resident command dispatcher to perform – something similar to the DOS print spooler. This makes it vital that you never use the ROBOT command from within the 'shell' feature of another application. The manual only casually mentions this problem, saying that you should try to write more comprehensive single scripts to avoid multiple copies of Robot being resident in memory, but in my humble opinion this is a serious flaw which reduces the usefulness of the program as a whole.

The remaining programs in the suite are called SNAPSHOT and REMEMBER. SNAPSHOT is another memory resident utility which you can use to capture text screens – graphics displays are not supported. To grab a screen, you simply press CTRL-SHIFT-Backspace and whatever is on the screen at the time is written to one of eight possible files. Once grabbed, the stored image can be used as a pattern – you can ask Robot to wait until the screen looks like this before it continues with the task in hand. It is not necessary for the entire screen to match the stored pattern – the script can be made to check only a portion of the display for a match if that is all that is required.

REMEMBER is another TSR program which is analogous to SNAPSHOT, except that this one is used to record keystrokes used to operate a program, ie the input to the program rather than the output. Once you know the keystrokes, these too can be pasted into a script – this is a little like the 'learn' mode in some comms packages. As far as I can tell, both of the above utilities assume that you will use the FRONT program to insert stored screen images and keyboard sequences into a script.

Documentation

The Robot manual is a 100 page affair, written primarily for the near beginner to computing. It has no index, but the contents pages are reasonably complete. Chapter one covers installation. It describes the contents of the disk and is very clear. The remaining chapters describe how to use the package and provide a complete list of script commands. I found much of the manual almost patronising in tone – the very first sentence in chapter two was the cringe-inducing 'Robot obeys your commands...'.

ISL have somewhat confused the target audience for this manual. By that I mean that, while the output of the package is intended to make life simpler for beginners, that beginner will only be a user of Robot scripts created by a more experienced person. There are simply too many things to get wrong for such a beginner to write scripts themselves. To put it plainly, if the person reading this manual needs to be instructed on the use of the Ctrl key on a PC keyboard (as the manual patiently explains) then that person is probably not the right one to be writing the scripts which others will use.

Usefulness

So, how useful is the Robot suite of programs as a software debugging aid? For this discussion, I need to split 'debugging' into two topics; debugging and quality assurance (or simply testing). I am not convinced that Robot has much of a role in software debugging. To find a bug with Robot, you would have to have a repetitive task which can be defined in such a way that Robot can be left to exercise that task until a bug appears (or hopefully doesn't). There is, I admit, a certain type of bug which is only found if you try a program feature n times – part of the general class of boundary condition bugs – however it seems that there are faster ways of finding such bugs, and that this form of testing is so limited as to be almost useless in my case. You may, of course, feel differently.

In my experience, the majority of bugs are easily found by the programmer, and the remainder are found by users (hopefully at the beta test stage) who try something that the programmer did not foresee, or experience conditions that the programmer did not provide for, such as an extreme shortage of disk space or memory. It seems to me that the worst way of looking for such bugs is to have the programmer define a series of Robot tasks to exercise components of the program in exactly the way the programmer expects them to be used.

If you have a simple repetitive task which you perform regularly, then Robot can very likely turn this into a single keystroke operation

If a programmer knows that a bug is there, he will examine the code or will get stuck in with a debugger. When the programmer thinks there are no bugs left then he releases the program for others to test. I find it difficult to see a suitable niche for Robot in this scenario.

However, I feel that the most promising possibility for Robot lies on the quality assurance side, ie the procedure described earlier of having a prepared suite of test scripts which exercises all or most of the functionality of a program, and which can be used as part of the final test phase to which you subject new versions of your own software. This probably is a valid use, but it has the obvious problem that new versions of the software may not be intended to function identically to previous ones (unless the new version is simply a bug fix), and so the test suite would have to be modified – however in this situation you would be changing your test procedure whatever the QA system adopted. One would also have to be very sure that using Robot did not itself introduce its own bugs, as TSRs are notorious for so doing. However, I have no reason for believing that the current release version of Robot is anything but reliable.

Conclusion

Robot is designed primarily for making life easier for the complete newcomer to the world of personal computing. In that role I have no doubt that it will serve very well (provided that a more experienced person such as a .EXE reader writes all the scripts). As a direct software debugging aid I feel that it is not quite so useful, and there are more specialised utilities which will serve the programmer better. It may be more useful as a hardware debugging aid, but I am sure that better tools exist in that area also. As a mechanised quality assurance tool, it may well take a lot of the grind out of that task – this is certainly the area which I feel holds the most promise for Robot. To me, Robot looks a little on the pricey side, especially considering the cost of individual run-time licences. However ISL point out that their nearest competitor is PC Automator, which is considerably more expensive.

.EXE

Don Milne is the author of Odyssey, a package that provides MNP-corrected communications on a PC. He can be contacted on 0224 63100. Robot requires a PC with 256K RAM, and DOS 2.x or later. Colour adds to the utility of the package but is not required. The Robot files occupy little more than 100 K of disk space, so you will probably not even require a hard disk. The single user developer's version, as reviewed, costs £395, and run-time licences are £95 one-off, with discounts for bulk orders. Contact Information Systems Ltd.on 0732 63881.

3+Open in Action

In the second of 3 instalments, Chris Adie continues his investigation into the merits of the 3+Open network. This month, he concentrates on security.

3+Open OS/2 workstations can be configured to use one of three types of logon security – centralised, distributed or none (the last one being the install-time default). Confusingly, 'no' logon security does not mean that all server resources are wide open to everyone. What it means is that you can issue the NET LOGON command, giving any username and password you like, without an error occurring. Only when you attempt to access a server will your username and password be checked against the server's user account database. When you issue, say, a NET USE command, the username and password stored in your workstation are encrypted and transmitted to the server along with details of your request. One benefit of this technique of logon validation is that if you are registered with the same username and password on more than one fileserver, you don't need to log onto each server individually to use its resources.

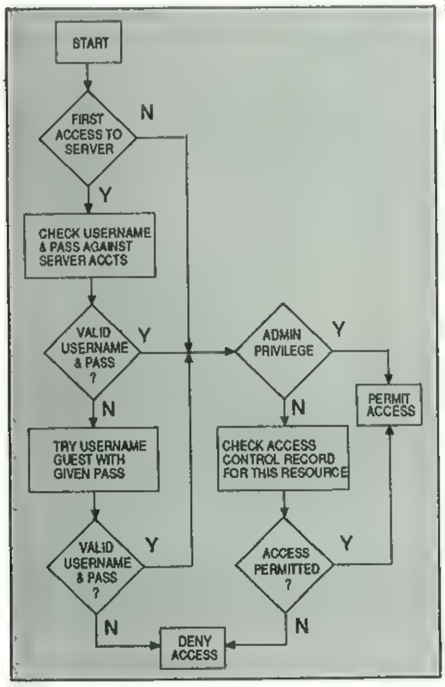


Figure 1 – Validating a resource access request

Figure 1 shows what happens when a workstation attempts to use a server resource. If it is the first access by the user to this server, his username and password are checked against the server's user account database. If the user is recognised, the server goes on to check whether he or she has permission to use the resource (by looking at the explicit or default access control record for the resource). If so, access is granted and the NET USE succeeds. If the username and password supplied are not valid, the server tries username 'GUEST' with the given password. If that succeeds, the access control record will be examined for GUEST's permissions, and access granted if permission exists. The GUEST account (with no password) is created during installation – you may wish to remove it to improve the security.

All usernames have a 'privilege' which can be one of Admin, User or Guest. Admin users (ie network administrators) have full access to all resources – they are not bound by the access control record mechanism. User privilege is the normal setting, while the GUEST username has Guest privilege. The only difference between Guest and User privilege is that Guest users are not members of the USERS group, so that you can exclude them from access to some (or all) resources.

Administration

The administrator can assign users to what are known as groups. A group can be assigned permissions (but not a privilege level) and appear in an access control record just like a username. When checking a username against an access control record, the server first looks for the username itself and, if it occurs, the indicated permissions are applied. Only if the username itself does not occur, does the server check to see if any groups to which the user belongs occur and, if so, it uses the logical OR of the permissions of all such groups. A consequence of this algorithm is that a group member can be assigned a reduced level of access to a resource compared to other members of the group. For instance, suppose that user PAT is a member of group MICROSUPPORT, which has RWCDAP access to a resource on the server. If PAT is allocated 'N' access to the same resource, he will be unable to use it although everyone else in the group can do so. This is the only use of the N permission mentioned earlier – it is pointless giving N permission for a resource to a group, for instance, since the effect is just the same as leaving the group out of the access control record altogether.

The resource access validation process shown in Figure 1 and described above takes place irrespective of the type of logon security selected. In 3+Open terms, having logon security means that the username and password are validated at the time they are typed in, rather than later when a NET USE command is issued. With centralised logon security, a NET LOGON command causes the given username and password to be sent to a server (nominated in the workstation's LANMAN.INI file) for validation against the server's user accounts database. With distributed logon security, a number of servers on the LAN are capable of validating logon requests, and a workstation can be configured to use one particular server, or to broadcast logon requests to allow any server to respond. (Passwords are always transmitted on the LAN in encrypted form). The NET LOGOFF command will terminate any NET USE links in effect, (prompting for confirmation first), and will then clear the stored username and password in the workstation.

It is quite straightforward for a user to bypass logon security (ie to select 'no' logon security) by editing his LANMAN.INI file. Logon security is, therefore, quite an inappropriate name for this feature of

520ST-FM SUPER PACK

The Atari Super Pack is ideal for you if you want to get off to a flying start with the best in entertainment software. The Pack includes a 520ST-FM with ½Mb RAM, a built-in 1Mb disk drive, over £450 of top games and a joystick. If you buy the Super Pack at Silica Shop, we will add our own ST Starter Kit (worth over £200), Free Of Charge. Return the coupon for details.

£399 INCLUDING VAT

1Mb DISK DRIVE £450 OF SOFTWARE

ARCADE GAMES

Arkanoid II	Imagine	£19.95
Beyond The Ice Palace	Elite	£19.95
Black Lamp	Firebird	£19.95
Buggy Boy	Elite	£19.95
Chopper X	Mastertronic	£9.99
Ikari Warriors	Elite	£14.95
Marble Madness	Electronic Arts	£24.95
Quadralien	Logotron	£19.95
Ranarama	Hewson Consultants	£19.95
Return To Genesis	Firebird	£19.95
Roadwars	Melbourne House	£19.95
Starquake	Mandarin	£19.95
Test Drive	Electronic Arts	£24.95
Thrust	Firebird	£9.95
Thundercats	Elite	£19.95
Wizball	Ocean	£19.95
Xenon	Melbourne House	£19.95
Zynaps	Hewson Consultants	£19.99

SPORTS SIMULATIONS

Eddie Edwards Super Ski	Elite	£19.95
Seconds Out	Tynesoft	£19.95
Summer Olympiad '88	Tynesoft	£19.95

PRODUCTIVITY SOFTWARE

Organiser	Triangle Publishing	£49.95

JOYSTICK

Atari CX40 Joystick	Atari Corp	[illegible]

FREE ATARI BUNDLE VALUE: £458.97

With SM124 mono monitor: £498 INC VAT — With SC1224 colour monitor: £698 INC VAT

ATARI ST

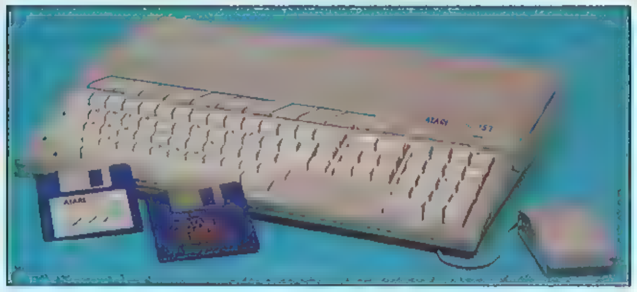
ST COMPUTERS

The range of Atari ST computers offers something for everyone. From the games enthusiast who wants the challenge of the very best in arcade action, to the businessman who wants to make financial forecasts or faultless presentations. The ST offers high quality graphics, sound and speed for the gamer, whilst providing a fast, user friendly and affordable solution to business. The ST is now firmly established in the home environment and boasts a wealth of users in education, local government, television, and a variety of different businesses. Software for the range stretches to cover applications as diverse as ENTERTAINMENT, ACCOUNTS, ART, COMMUNICATIONS, COMPUTER AIDED DESIGN, DATABASES, DESKTOP PUBLISHING, EDUCATION, MUSIC, PROGRAMMING, SPREADSHEETS, WORD PROCESSING and more. For a full list of the software available, as well as details of the ST range, complete and return the coupon below.

All prices correct at the time of going to press. E&OE

520ST-FM EXPLORER PACK

WITH BUILT-IN 1Mb DISK DRIVE



The value for money offered by the Atari ST range is reflected in the Explorer Pack featuring the 520ST-FM computer with 512K RAM. The 520ST-FM computer now comes with a built-in 1 Mb double sided disk drive as well as a free mouse controller and a built-in TV modulator. The new 520ST-FM Explorer Pack includes the 520ST-FM computer, the arcade game Ranarama, a tutorial program and some useful desktop accessories. In addition, if you buy the Explorer Pack from Silica, we will give you the Silica ST Starter Kit worth over £200, FREE OF CHARGE. Return the coupon for details of our Starter Kit and of the full ST range.

£260 + VAT = £299

ONLY £2.51 PER WEEK — RETURN COUPON FOR DETAILS

+ SM124 mono monitor: £398 INC VAT — + SC1224 colour monitor: £598 INC VAT

1040ST-FM PROFESSIONAL PACK

NOW WITH TV MODULATOR

For the serious home user and the small business, we are pleased to announce a new package based around the 1040ST-FM. The 1040ST-FM has 1Mbyte RAM and a 1Mbyte built-in disk drive. In addition, the 1040ST-FM now comes with a TV modulator built-in. (The previously available 1040ST-F was designed for use with a monitor only and did not come with a modulator.) This modulator allows the 1040ST-F to be plugged directly into any domestic TV set, and comes complete with a lead to allow you to do so. The new 'Professional Pack' from Silica includes the new 1040ST-FM with modulator plus four high quality software packages including a spreadsheet, database, word processor and programming language. This 'Professional Pack' software will enable you to get straight down to business with your new computer. In addition to this software (worth £384.84), if you buy the Professional Pack from Silica Shop, you will also receive the Silica ST Starter Kit (worth over £200), Free Of Charge. Return the coupon for further information.

£499 INCLUDING VAT

ATARI 1040ST-FM	(Computer)	£499.99
VIP PROFESSIONAL	(Spreadsheet)	£149.95
MICROSOFT WRITE	(Word Processor)	£149.95
SUPERBASE PERSONAL	(Database)	£59.95
BASIC DISK & MANUAL	(Language)	£24.98
	NORMAL RRP:	£884.82
	LESS DISCOUNT:	-£385.82
	PROFESSIONAL PACK PRICE:	£499.00

With SM124 mono monitor: £598 INC VAT
With SC1224 colour monitor: £798 INC VAT

2Mb & 4Mb MEGA ST

The MEGA ST computers are styled as as lightweight keyboard with a separate CPU, connected by a coiled telephone style cable. There are two versions of the MEGA ST, one with 2Mbytes of RAM and the other with 4Mbytes. Each version has a 1Mbyte double sided disk drive built-in to the CPU unit. The MEGA ST's do not come with modulator built-in and must therefore be used with a monitor. With every MEGA ST purchased, we will add the 'Professional Pack' software (worth £384.83) detailed above, plus the Silica ST Starter Kit (worth over £200) both Free Of Charge. Return the coupon for further details.

2Mb MEGA ST £899 INC VAT
+ mono monitor = £998
+ colour monitor = £1198

4Mb MEGA ST £1199 INC VAT
+ mono monitor = £1298
+ colour monitor = £1498

DTP PageStream £149 +VAT =£171.35

Desktop Publishing (DTP) is one of the fastest growing applications for personal computers. We are pleased to announce a powerful low cost package for the Atari ST called PageStream. PageStream costs only £149 (+VAT=£171.35) and, because it works with an Atari 1040ST and a Seikosha SP-180AI printer, you can be up and running with a complete system for less than £1000. Some of the features of PageStream are listed to the right. If you would like further information on this program, complete and return the coupon below, ticking the 'DTP' box in the corner.

- TEXT-FLOW AROUND GRAPHICS
- ROTATION OF TEXT & GRAPHICS
- SLANT OR TWIST ANY OBJECT
- POSTSCRIPT COMPATIBLE
- TAG FUNCTION
- AUTO/MANUAL KERNING & HYPHENATION
- GROUPING OF OBJECTS

WHY SILICA SHOP?

Before you decide when to buy your new Atari ST computer, we suggest you consider very carefully WHERE you buy it. There are MANY companies who can offer you a computer, a few peripherals and the top ten selling titles. There are FEWER companies who can offer a wide range of products for your computer and expert advice and help when you need it. There is ONLY ONE company who can provide the largest range of Atari ST related products in the UK, a full time Atari ST specialist technical helpline and in-depth after sales support, including free newsletters and brochures delivered to your door for as long as you require after you purchase your computer. That one company is Silica Shop. We have been established in the home computer field for ten years with an annual turnover in excess of £8 million and can now claim to meet our customers requirements with an accuracy and understanding which is second to none. But don't just take our word for it. Complete and return the coupon below for our latest literature and begin to experience the Silica Shop specialist Atari service.

SILICA STARTER KIT: Worth over £200. FREE with every Atari ST computer bought from Silica

PROFESSIONAL PACK: Free business software with 1040ST-FM and MEGA ST's bought from Silica

DEDICATED SERVICING: 7 full-time Atari trained staff with years of experience on Atari servicing

THE FULL STOCK RANGE: All of your Atari requirements from one place

AFTER SALES SUPPORT: The staff at Silica are dedicated to help you get the best from your ST

FREE CATALOGUES: Mailed direct to your home as soon as we print them, featuring offers as well as all of the new releases

FREE OVERNIGHT DELIVERY: On all hardware orders shipped within the UK mainland

PRICE MATCH PROMISE: We will match competitors on a same product same price basis

FREE TECHNICAL HELPLINE: Full time team of Atari technical experts always at your service

FREE SILICA STARTER KIT WORTH OVER £200

WITH EVERY ST - RETURN COUPON FOR DETAILS

ALL PRICES QUOTED INCLUDE FREE UK DELIVERY

DO YOU OWN AN ATARI ST?

If you already own an Atari ST computer and would like to be registered on our mailing list as an ST user, let us know. We will be pleased to send you copies of our price lists and newsletters FREE OF CHARGE as they become available. Complete the coupon and return it to our Sidcup branch and begin experiencing a specialist ST service that is second to none.

SILICA SHOP:

SIDCUP (& Mail Order) 01-309 1111
1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX
OPEN: MON-SAT 9am - 5.30pm LATE NIGHT: FRIDAY 9am - 7pm

LONDON 01-580 4000
52 Tottenham Court Road, London, W1P 0BA
OPEN: MON-SAT 9.30am - 6.00pm LATE NIGHT: NONE

LONDON .... 01-629 1234 ext 3914
Selfridges (1st floor), Oxford Street, London, W1A 1AB
OPEN: MON-SAT 9am - 6.00pm LATE NIGHT: THURSDAY 9am - 8pm

To: Silica Shop Ltd, Dept .EXE 9/89 1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX

PLEASE SEND FREE LITERATURE ON THE ATARI ST

Mr/Mrs/Ms. Initials: .... Surname: .... .EXE 12/89

Address:

Postcode

Do you already own a computer
If so, which one do you own?

DTP ☐

CIRCLE NO. 724

Figure 2 – 3+Open interface menus, with administrator-only options in bold

View	Message	Config	Status	Accounts
Network Servers	Send	Logon	**Device Status**	**Users/Groups**
This Workstation	Receive	Logoff	**Session Status**	**File Permissions**
Print Queues	Log File	Load Profile	**Opened Files**	**Other Permissions**
Comm Queues	Alias	Save Profile	Workstation Statistics	
This Server		**Server Options**	**Audit Trail**	
Other Server		Change Password	Error Log	
Exit		Stop Net Services		

3+Open! Of course, the resource access validation procedure (Figure 1) still applies, so bypassing logon security does not create a security breach. In fact, DOS workstations cannot use logon security. What, then, is the benefit of it? First, it gives a user instant feedback as to whether he has typed his password correctly. Second, it is the mechanism for implementing logon scripts. The Administrator can nominate a file (on the server which validates the logon request) as a user's logon script. This can be a batch file (.CMD), an executable program (.EXE) or a LAN Manager profile (.PRO).

A LAN Manager profile can contain NET SHARE, NET USE, NET PRINT and NET COMM commands. A .PRO file is created by the NET SAVE command, while the NET LOAD command will re-establish the SHARES, USES, PRINT and COMM queue settings stored in a nominated profile. There is a security loophole in 3+Open, concerned not with the server but with sending messages. The NET SEND command will transmit a message to appear as a pop-up on the screen of another workstation. Broadcasting messages to all workstations is also possible. It is not necessary to be logged on to do either of these, and it is trivial to set up a batch file which constantly pumps out irritating messages to every workstation on the LAN. This loophole must be plugged if LAN Manager is to find general acceptance as a serious LAN operating system.

Administering the Server

An OS/2 workstation user will probably tend to use the command line interface for most network-related tasks. However, it is more convenient to administer the server using the full-screen interface invoked by the NET ADMIN command. (Non-admin users can invoke a subset of this interface by giving the command NET on its own). The interface uses character-based windowing technology – it won't run in a Presentation Manager window, but requires a screen group all to itself. Although claimed to be SAA-compliant, there are many detailed points of difference between this user interface and the Common User Access of SAA. In fact, the only really SAA-compliant programs I've seen all come from IBM. An EGA or VGA colour display and a mouse are needed to make best use of the NET ADMIN program. Figure 2 shows the structure of the pull-down menus. Once you have mastered the basics of LAN Manager, the interface is easy to use, but it tends to be confusing if you plunge straight in, partly because there are sometimes several ways of doing the same thing.

As well as the administrator's and ordinary user's versions of the full-screen interface, there is a third version, invoked by the NET CONSOLE command, which is intended for use at an unattended server. Its main function is to allow printers and printer queues to be controlled by users who come to the server to collect their printer output. You cannot exit NET CONSOLE without giving a password, nor can you start up a session running the OS/2 command interpreter without logging on first. I have discovered that this restriction is quite easy to bypass, so here is a second security loophole for Microsoft and 3Com to plug. However, it is good practice to keep the file server in a locked room if there is any possibility of interference. If the server does not hold programs or data but is used purely as a print server, then of course this precaution is much less important.

A server running NET CONSOLE is in effect a dedicated server but, because of the multitasking nature of OS/2, concurrent server operation is the usual situation. OS/2 programs (and indeed DOS programs, should you be brave enough to enable the DOS box) can be run on the server. However, in order for such programs to participate in the resource access control mechanisms and benefit from printer spooling, it is necessary to load a special 'loopback' driver and to issue appropriate NET USE commands. There is apparently no way to prevent a server user from accessing the server's disks directly, bypassing the access control. This is another argument for keeping the server locked away, rather than using it as a workstation.

There are sometimes circumstances which make it impractical to lock away the fileserver, and Novell Netware with its non-standard disk format (which defeats attempts to access data by booting the server under DOS) offers significantly better security in such a situation, compared to 3+Open running on a PC. However, it is open to Microsoft to provide an installable file system for LAN Manager which would, together with a better NET CONSOLE program, offer the same level of security as Netware even when the server cannot be locked away. (The best option is still to lock the server up if you can – the software can't stop someone just switching the machine off.)

Log Files

A 3+Open server keeps two log files of interest to the administrator – an error log

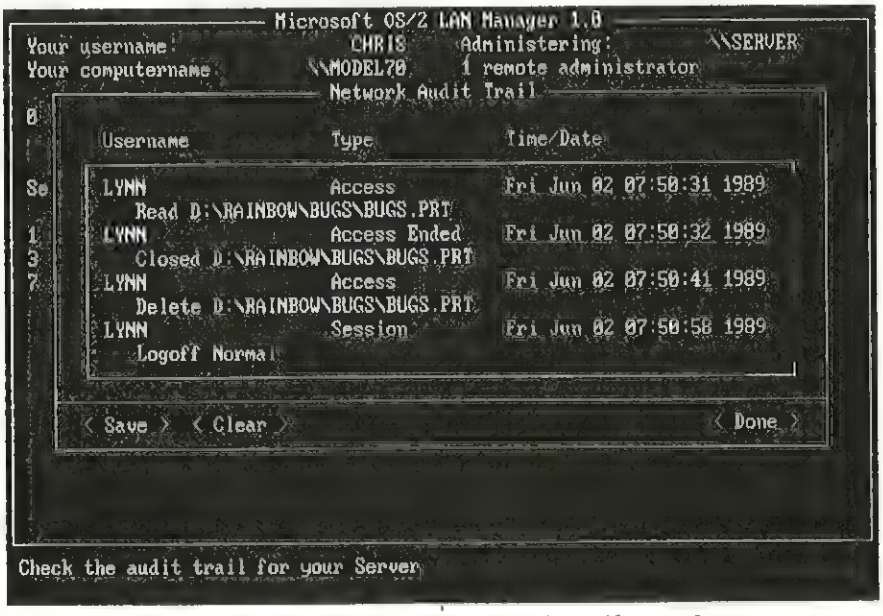


Figure 3 – The Audit Trail file produced by 3+Open

and an audit trail. The error log contains information on any problems which have occurred during network operation. The date, time, error number and the name of the program generating the error are stored, and the message text corresponding to the error number can also be displayed. The audit trail file (Figure 3) contains information about user logons and logoffs, and about the usage of resources. There is a special 'audit' bit in each access control record on the server, which controls the collection of audit information for the corresponding resources. This selectivity in controlling auditing is necessary to avoid generating a huge audit trail file. An entry is made in the audit trail only when a file is opened or closed, rather than for each individual read and write activity. Both the audit and error log files can be restricted in size to avoid wasting space.

Backing up the server is an extremely important part of network management, and if you are running 3+Open on 3Com's 3S401 or 3S402 dedicated fileservers with built-in tape streamer, there is no problem. Backups of the hard disk partitions can be scheduled for automatic execution late at night if required, and you can back up workstations' hard disks as well. However, if you are using an ordinary PC/AT or PS/2 as a server, 3+Open provides very little to help you back it up. There is a command scheduler called AT.EXE, allowing you to run a program or batch file at some later time, which you could use to schedule a backup operation, but few tape streamers currently available provide backup software which will run in OS/2 protected mode. There is clearly a market opportunity here for the first tape streamer manufacturer to offer OS/2 support.

If the server happens to crash or is powered off, any NET USE sessions active are marked as disconnected by the workstations, and any open files are forced closed. When the server comes back on line, the NET USE sessions are automatically re-established. Incidentally, 3Com sell a separate program for 3+Open called LAN Secure, which offers extra security facilities to the administrator. Another program, LAN View, makes network problem diagnosis easier.

DOS Workstations

Only a limited subset of 3+Open facilities are available to DOS workstations. Server disk and printer resources can be used from a DOS workstation, but not communications queues. The NET command does not provide a full-screen interface under DOS, so 3Com supply the Microsoft 'DOS Manager' program, as illustrated in Figure 4. This is an SAA-style shell which lets you perform simple file management and networking tasks.

The most significant limitation of a DOS workstation is that you can't administer the server from it. NET ADMIN must be run at the server itself, or from an OS/2 workstation. If the network covers mainly DOS machines, it may be necessary to budget for a machine configured for OS/2 specifically for this purpose, particularly if a 3Com 3S40x dedicated fileserver is to be used (since it has no keyboard or display). On the other hand, this restriction does limit the possibilities for hacking.

The other major restriction on DOS workstations under the current version of 3+Open is the amount of memory required. A 360 K DOS system disk with just COMMAND.COM and the 3+Open DOS software on it is nearly full. On an 8086 or 80286 PC, the 3+Open software takes up a staggering 166 K of RAM, leaving only 414 K for DOS applications. The situation is slightly better on some 80386 machines, because 3Com supply software for loading drivers into spare memory above the 640 K limit. On a PS/2 Model 70 this reduces the memory usage to 102 K. The amount of memory used also depends slightly on whether you select the XNS or DLC protocol – DLC 'only' takes 156 K.

3Com, acknowledging that this memory overhead is a serious defect in their LAN Manager strategy, have announced a new protocol called NBP (NetBios Plus protocol) which requires very much less memory in a DOS workstation. A pre-release version of NBP is now available free of charge to existing 3+Open customers. On a DOS workstation, NBP occupies about 83 K – roughly half of what the XNS protocol stack takes up – leaving 516 K for user applications. 3Com say an extra 30 K of RAM can be saved by loading drivers into extended memory on a 286 or 386 machine, but the required HIMEM.SYS driver was not supplied with my early copy of NBP. The impact on the DOS box in an OS/2 workstation is similar – 443 K free instead of 355 K under XNS.

Although login security is not available under DOS, the NET LOGON command does provide similar facilities. After prompting for your user name and password, it looks for an environment variable called HOME-SERVER, and attempts to link drive H: to the USERDIRS sharename on that server. (The server name, drive letter and sharename can alternatively be specified on the command line.) This ensures that the logon is validated by the nominated server – it is therefore very similar to the centralised logon security feature available to OS/2 workstations. However, no explicit logon script facility is provided for DOS workstations – batch files are used instead.

There are one or two minor annoyances with 3+Open on a DOS workstation. Error messages can be misleading – for instance the message 'Unexpected network error' usually means that you have tried to NET USE a server resource before logging on. Although a DOS workstation can receive messages sent by OS/2 workstations or the server using NET SEND, it can't send messages itself. EXE

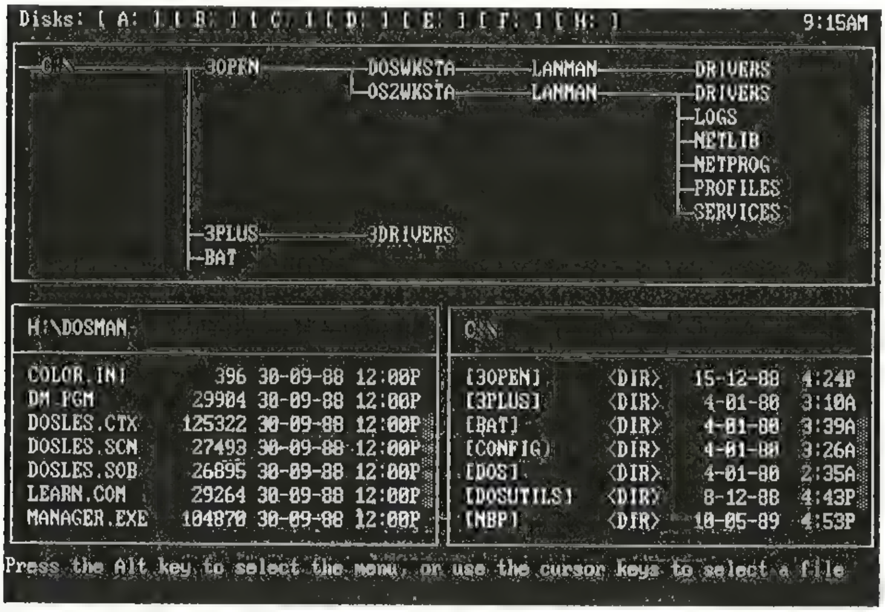


Figure 4 – The Microsoft DOS Manager

Chris Adie is with the University of Edinburgh Computing Service. The final part of this trilogy will appear in February (.EXE does not publish a January issue).

Books

This month reviews of two unusual volumes: a book of interviews and a book of pitfalls.

Programmers at Work (2nd edition)

They're all here: there's Charles Simonyi, dear old Charles; Bill 'William' Gates, dear Bill, he looks so young doesn't he? There's Wayne 'dBASE' Ratliff, dear, dear Wayne. 'Programmers at Work' consists of 19 lengthy interviews with the World's Greatest Living Programmers (all men, natch). The front cover shows four of them, drawn in a style so evocative of the tough, American way that one would be unsurprised to find John Wayne, all dressed up in his True Grit togs, right there beside the aforementioned Wild Bill. Looks like we are in for a shower of very American-style eulogy such as would make even Mickey Mouse's toes curl.

Actually, not so. Susan Lammers's line of questioning is careful, but not aggressive – rather like the late Roy Plomley on 'Desert Island Disks'. Some of her interviewees appeared to need little prompting. A case in point is one Jef Raskin, creator of the Macintosh project at Apple. His dislike of Apple executive Steve Jobs is very evident:'I was amused by a recent article where he [Jobs] said, "I have a few good designs in me still." He never had any designs. He has not designed a single product.' Well, miaow, miaow.

Another person that I warmed to was Toru Iwatani, the Japanese designer of the game 'Pac Man'. (Yes you do. Think back to wasted evenings spent in pubs in the early eighties.) When Ms Lammers asks, *as the very first question of the interview,* 'How did you first become interested in computers?', Iwatani comes straight back with 'I must tell you, I don't have any particular interest in them.' Given the title of the book, it is much to Ms Lammers's credit that she then spins the interview out for five more interesting pages. Myself, I would have said 'thank you very much for your time' and left.

The main interest of the book comes from the little personal histories and anecdotes that emerge from the interviews. There are several variations on the theme of 'we set up at high school, using a toilet as an office, placed a quarter page advertisement in Byte and three months later were millionaires.' Reassuringly, for those of us who lack the entrepreneurial spark, some of them go on to meet their come-uppance; notably Dan Bricklin, who devised VisiCalc and is now probably best known for the appallingly-named Dan Bricklin's Demo Program. His company, Software Arts, got sued by the marketing company VisiCorp.

The interviewees' Opinions about Things are less enthralling. On the programming front, there seems to be a consensus among the great and the good that a) someone else's commenting style is his own business, but if you monkey around with his code then you should try to imitate it and b) the maximum practical size for a programming team, whatever the project, is six. Are you listening to this, you Star Wars-mongers? As for the nuggets dropped concerning the future of computers and the direction of AI, these seemed no more profound or perceptive than those which could be canvassed from 19 common souls, picked out at random while wandering around Woolworth's doing their Saturday morning shopping.

If I have a criticism of the excellent Ms Lammers, and it is only a friendly warning from one who frequently falls into the same trap, it is that she slightly overdoes her sub-Chandler style in the introductions. 'It was pouring with rain, and the recently installed glass roof covering the three-storey, open lobby was leaking in several places. A short man with longish brown hair, dark-rimmed glasses and carrying an umbrella, entered the lobby doors. It was Frankston. I struck a match on my thumb-nail and inhaled deeply on a Camel.' Ok, so I made up the Camel, but you see what I mean.

In conclusion, then, this book is a lot of fun and, at £7.95, pretty good value. It also answers the Gates question that everybody who sees his photograph asks: 1955. He must have a Dorian Gray style piece of software in his attic, which ages rapidly while he remains young. Ah yes, I remember now, MS-DOS.

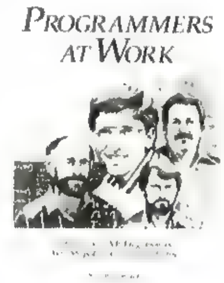
Author: Susan Lammers

Publisher: Tempus (Microsoft Press)

Price: £7.95

ISBN: 1 55615 211 6

Pages: 391



C Traps and Pitfalls

I must admit I cheated on this one: I read the PCW review of this book (November '89 issue) before writing this. A little unsporting, but then it is Christmas. 'An invaluable book', gushes the reviewer, 'to every C programmer.' I beg to differ. My conclusion, which I will give you now, so that you don't have to trail all the way to the bottom of the piece, is that it is a modest book of use to the inexperienced.

C Traps and Pitfalls is divided into eight chapters plus an introduction and an appendix. Each chapter covers one area of C which may cause pitfalls: lexical pitfalls, syntactic pitfalls and so on. To give you an idea of the level of the book, here is the complete list of lexical pitfalls: = is not ==, & and | are not && or ||, greedy lexical analysis (ie if you leave out spaces then you deserve what you get, as in `x/*p` is not the same as `x / *p`), integer constants (beginning with 0 are octal) and strings and constants (are different).

'But', I hear you cry, 'I knew all that anyway.' Of course you did, you have spent a little time programming in C. The problem with this book is that, although it made a sweet little in-house paper (which is how it originated at AT&T Bell Labs; bug contributors include the venerable B Stroustrup himself), there isn't nearly enough material and it is too lightweight. The pitfalls listed here would fit nicely onto about two sides of typed A4, so that when one hit a bug, one could glance through it like a check-list.

The reason that I am being a little hard on this book is that I think that it is a missed opportunity. I think that if the author included a summary of pitfalls (in a check-list form ordered both by logical group and frequency of occurrence) and had many, many more examples (perhaps 200), then it could be rather good.

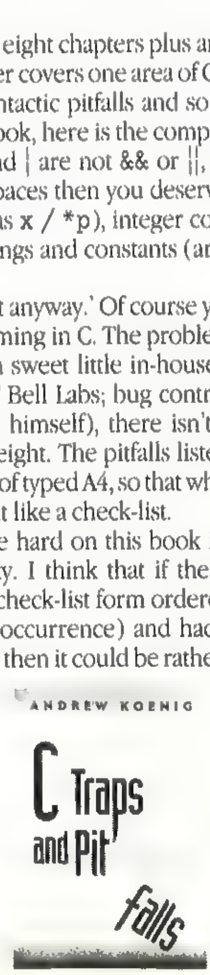
Author: Andrew Koenig

Publisher: Addison-Wesley

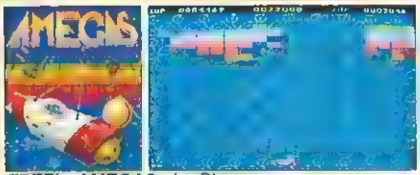
Price: £13.49

ISBN: 0 201 17928 8

Pages: 147



FREE! TENSTAR PACK WORTH OVER £229!



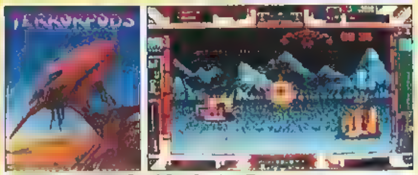
FREE! - AMEGAS - by Players

FREE! - INSANITY FIGHT - by Microdeal

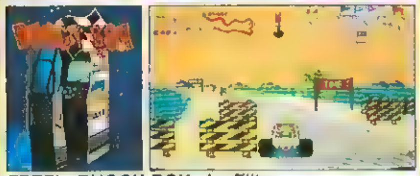
FREE! - ART OF CHESS - by SPA

FREE! - MERCENARY COMP - by Novagen

FREE! - BARBARIAN, ULT WARRIOR - by Palace



FREE! - TERRORPODS - by Psygnosis



FREE! - BUGGY BOY - by Elite

FREE! - THUNDERCATS - by Elite

FREE! - IKARI WARRIORS - by Elite

FREE! - WIZBALL - by Ocean

Commodore AMIGA

£346.95 +VAT **£399**

INCLUDES FREE UK DELIVERY

The Amiga 500 is one of a new breed of technologically advanced computers, which are now emerging as the new standard for home computing, based around the new Motorola 68000 chip. The A500 has 512K RAM and a 1Mbyte double sided disk drive built-in. It can be connected directly to a wide range of monitors, or to a domestic TV set through a TV modulator. Designed with the user in mind, the A500 features a user friendly WIMP environment and comes supplied with a free mouse. And, when you buy your Amiga from Silica Shop, the UK's No1 Amiga specialists, you will experience an after sales service that is second to none, including a technical support helpline and free newsletters and price lists. Return the coupon below for our current information pack, which will give details of the Silica service and the very latest Silica Amiga offers. E&OE

WHY SILICA SHOP?

Before you decide when to buy your new Commodore Amiga computer we suggest you consider very carefully WHERE you buy it. There are MANY companies who can offer you a computer, a few peripherals and the top ten selling titles. There are FEWER companies who can offer a wide range of products for your computer as well as expert advice and help when you need it. There is ONLY ONE company who can provide the largest range of Amiga related products in the UK, a full time Amiga specialist technical helpline and in-depth after sales support, including free newsletters and brochures delivered to your door for as long as you require after you purchase your computer. That one company is Silica Shop. We have been established in the home computer field for ten years and can now claim to meet our customers requirements with an accuracy and understanding which is second to none. Here are just some of the things we can offer you.

THE FULL STOCK RANGE: The largest range of Amiga related peripherals, accessories, books and software in the UK.

AFTER SALES SUPPORT: The staff at Silica Shop are dedicated to help you to get the best from your Amiga.

FREE NEWSLETTERS: Mailed direct to your home as soon as we print them, featuring offers and latest releases.

FREE OVERNIGHT DELIVERY: On all hardware orders shipped to Silica Shop customers within the UK mainland.

PRICE MATCH PROMISE: We will normally match our competitors offers on a 'same product same price' basis.

FREE TECHNICAL HELPLINE: Full time team of Amiga technical experts to help you with your technical queries.

But don't just take our word for it. Complete and return the coupon below for our latest Amiga literature and begin to experience the Silica Shop specialist Amiga service.

PHOTON PAINT PACK

A500 Computer	£399.99
TV Modulator	£24.99
Photon Paint	£69.95
TenStar Pack	£229.50
TOTAL RRP:	£724.43
LESS DISCOUNT:	£325.43
PACK PRICE INC VAT:	**£399**

1084S MONITOR PACK

A500 Computer	£399.99
1084S Colour Monitor	£299.99
Photon Paint	£69.95
TenStar Pack	£229.50
TOTAL RRP:	£999.43
LESS DISCOUNT:	£350.43
PACK PRICE INC VAT:	**£648**

FREE FROM SILICA

When you buy the Amiga 500 from Silica Shop, you will not only get a high power, value for money computer, we will also give you some spectacular free gifts. First of all, we are now including a TV modulator with every A500 stand alone keyboard, so you can plug your Amiga straight into your TV at home (the modulator is not included with the A500+A1084S pack as it is not required for use with monitors). Secondly, we have added a free copy of Photon Paint, an advanced graphics package with an RRP of £69.95. Last (and by no means least!), so that you can be up and running straight away, we are giving away the sensational TENSTAR GAMES PACK with every A500 purchased at Silica Shop. This pack features ten top Amiga titles which have a combined RRP of nearly £230! Return the coupon for details.

FREE TENSTAR PACK

When you buy your Amiga 500 from Silica Shop, we want to make sure you get the best deal possible. That is why we are giving away the TENSTAR GAMES PACK worth nearly £230, absolutely FREE with every A500 purchased from us. The TenStar Games Pack includes ten titles for the A500, each individually packaged in its own casing with instructions.

Amegas	£14.95
Art Of Chess	£24.95
Barbarian, Ult Warrior	£19.95
Buggy Boy	£24.95
Ikari Warriors	£24.95
Insanity Fight	£24.95
Mercenary Comp	£19.95
Terrorpods	£24.95
Thundercats	£24.95
Wizball	£24.95
	£229.50

TOTAL RRP: £229.50 INC VAT

DO YOU OWN AN AMIGA?

If you already own an Amiga computer and would like to be registered on our mailing list as an Amiga user, let us know. We will be pleased to send you copies of our price lists and newsletters FREE OF CHARGE as they become available. Complete the coupon and return it to our Sidcup branch and begin experiencing a specialist Amiga service that is second to none.

SILICA SHOP:

SIDCUP (& Mail Order) 01-309 1111
1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX
OPEN: MON-SAT 9am - 5.30pm LATE NIGHT: FRIDAY 9am - 7pm

LONDON 01-580 4000
52 Tottenham Court Road, London, W1P 0BA
OPEN: MON-SAT 9.30am - 6.00pm LATE NIGHT: NONE

LONDON 01-629 1234 ext 3914
Selfridges (1st floor), Oxford Street, London, W1A 1AB
OPEN: MON-SAT 9am - 6.00pm LATE NIGHT: THURSDAY 9am - 8pm

To: Silica Shop Ltd, Dept .EXE 9/89 1-4 The Mews, Hatherley Road, Sidcup, Kent DA14 4DX

PLEASE SEND ME FREE LITERATURE ON THE AMIGA

Mr/Mrs/Ms: Initials: Surname:

Address:

Postcode:

Do you already own a computer
If so, which one do you own?

CIRCLE NO. 727

X-Window System* Information

As the European centre of X expertise IXI Limited will help you exploit this important software standard.

BOOKS & MANUALS

Specialist books for Software Developers.

Reference Materials, O'Reilly Programming Tutorials for Xlib and Toolkits, Users' Guide to X.

SOURCE CODE

The latest release of the MIT source code, modified by IXI to load on System V as well as BSD UNIX.

Available on 0.25″ cartridge tape and IBM PC-AT 5.25″ floppies (UNIX tar files).

TRAINING

Our well established training courses cover:

1 day Strategic Overview of X.

4 days Programmer's Workshop with hands-on training.

0223 462131
+44 223 462131

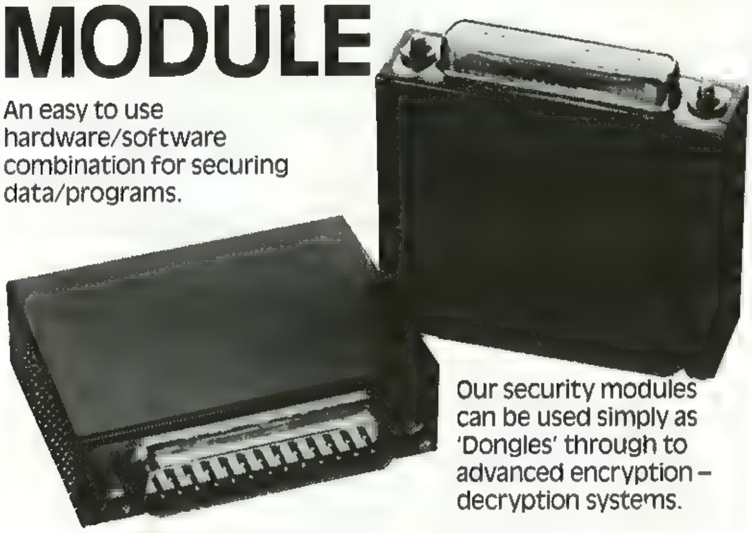
IXI Limited
62–74 BURLEIGH STREET
CAMBRIDGE CB1 1OJ
ENGLAND

*X-Window System is a trademark of MIT

CIRCLE NO. 728

SOFTWARE SECURITY MODULE

An easy to use hardware/software combination for securing data/programs.



Our security modules can be used simply as 'Dongles' through to advanced encryption – decryption systems.

- ★ For IBM PC family (and compatibles)
- ★ Variants for other systems
- ★ Advanced construction – rugged and compact
- ★ Proven throughout the world
- ★ Transparent operation – user friendly
- ★ Driver software supplied for variety of applications

CTL

Control Telemetry of London
11 Canfield Place, London NW6 3BT
Tel: 01-328 1155 Fax: 01-328 9149

CIRCLE NO. 729

S–GKS portable graphics toolkit

- written in 'C' for portability & performance
- available on a wide range of systems from PC's to mainframes
- support for the popular workstations from SUN, Apollo, IBM, Hewlett-Packard, Sony DEC (VMS & Ultrix including DecStation)
- full GKS level 2b functionality on PC's
- Fortran and 'C' language support
- ISO certified

new features:

- X-Windows driver
- CGM (Computer Graphics Metafile) support

GKS Training Courses

- 5-day training course on developing interactive graphics programs using GKS (courses run for both Fortran and 'C' users)

SSL

Scientific Software Limited
Rose Industrial Estate
54 Marlow Bottom
Marlow, Bucks SL7 3ND
Tel. 0628 890011

CIRCLE NO 730

.EXE IN REAL TIME

THE SOFTWARE THAT DRIVES THE TRAIN THAT CARRIES THE YUPPIES

CROSS-COMPILING FOR MULTIPLE TARGETS - HOW HP ALWAYS PRODUCES THE BEST CODE

REAL TIME OPERATING SYSTEMS - WHAT THERE IS AND WHAT IT DOES

GOING HIGH-LEVEL - HOW ONE COMPANY MADE THE BREAK FROM ASSEMBLY TO C

You don't design your own silicon, why write your own real-time kernel?

Engineers no longer build computers from discreet transistors; they use fast, reliable and inexpensive integrated circuits. Why then should programmers continue to create real-time software – one instruction at a time?

Writing your own kernel is not only expensive and resource intensive – it only addresses today's *known* requirements. For many years Software Components Group have been providing companies worldwide with **software building blocks**, freeing programmers from tortuous development cycles and never-ending maintenance schedules, allowing them to apply their genius where it really counts – at the application end.

pSOS™, SCG's multi-tasking, real-time kernel, offers unparalleled levels of performance, functionality and ease of use. In 68K (including 68030) embedded applications, **pSOS+** has no equal. Code calls to the kernel are simple to make (via a compiler interface) and fully deterministic. The function set provided is ideally suited to a vast range of real-time applications. In addition, **pSOS+** (together with the peripheral modules) is completely target-hardware independent.

Multiprocessing

In multiprocessing systems, integration and debugging can very often be a nightmare. Unless of course, you have sophisticated tools which are tailored to your needs (and those of your kernel). **pROBE+™** and **XRAY+™** provide such tools – the former presents an intelligent window for interprocess diagnostics, the latter provides a truly user-friendly interface into source code debugging. Together they represent an extremely powerful debugging system which allows rapid and thorough integration and testing.

Tomorrow

Your future effectiveness is completely dependent upon your continuing investment in software maintenance; in-house produced kernels will invariably drain resources for as long as you take such products to market. The **pSOS+** range is designed to eliminate this wastage – on-going **pSOS+** developments will enable you to grow your competitive advantage whilst still enhancing performance and reliability.

As with the proverbial oak tree, future growth depends upon laying down solid roots today. Can you really afford to wait until tomorrow? Find out more by contacting Geoff Revill at Thame Microsystems* on 084 421 7272.

THAME MICROSYSTEMS LTD.

A Member of the Memec (Memory and Electronic Components) PLC Group

THAME PARK ROAD, THAME, OXON OX9 3UQ
TELEPHONE: 0844 261456. TELEX: 837508
FAX: 0844 261682

* Thame Microsystems are the only approved **pSOS+** Distributor in the UK.

pSOS+ and **pROBE+** are registered trademarks of Software Components Group **XRAY** is a registered trademark of Microtec Research Inc.

CIRCLE NO. 731

Contents

EDITOR : WILL WATTS
ADVERTISING: SANDRA INNISS-PALMER
PRODUCTION: KATERINA ADAMS
RACHEL AFFLECK

.EXE REAL TIME SUPPLEMENT

Editorial

What is a real time system? This sounds like the sort of clever-clever question that the lecturer asks at the beginning of a real time software course. Indeed, I first heard it asked and answered while attending such a course, which was given at a special college maintained by a well-known defence manufacturer. The specimen answer, which the lecturer attributed to Michael Jackson, went as follows: 'A real time system is a system whose output is capable of affecting its input.'

This definition stands up to almost half a second's thought. A weather monitoring radar system, which updates animated images of rain clouds twice a second, is not, by these lights, a real time system. Whatever the output it produces, clearly it is not going to affect the clouds (I'm ignoring, for the purpose of my argument, the possibility of the computer being able to fire rainfall-inducing laser beams into the sky) and therefore its input is independent of its output. On the other hand, take a classical non-real time application; what about a payroll program that runs as a batch job once a month? If you do not think that its output (printed cheques) cannot affect its input (list of names and salaries), try reducing everyone's pay to £0000.00; then stand back and await developments.

Since attending the course, I've developed the suspicion that our lecturer made some sort of mistake. Perhaps she got the wrong Michael Jackson, and the definition above was given during a recording break in *Thriller*. Perhaps she got her text confused with Tom Lehrer's definition of Life ('Life is like a sewer: what you get out of it depends on what you put into it'). So I started to look for a more useful definition.

CS French's popular text *Computer Science - an instructional manual* says: 'Real time operation is operation on a unit time scale, ie on the same time scale as the simulated events.' A bit vague, that. Doesn't really exclude our batch cheque-issuing program, which runs on the same unit time scale as employees salaries: once a month. *Bong!* Wrong answer, thanks for playing. In his book *Introduction to Real-time Software Design*, ST Allworth writes '... many attempts have been made to define a "real-time" system ... these definitions are incomplete ... instead of attempting a definition, we shall discuss the properties and facilities that are commonly regarded as making up a real-time system.' Talk about cop-out city, Cincinnati.

I was quite taken by this effort, which I found in an Intel brochure: 'Real time: where a computer is used to record/control an external environment and has **absolute time constraints** that it **must meet without fail**.' The last bit perhaps betrays a desire to sell the Intel operating systems (which turn out, according to the brochure, to be red hot at meeting absolute time constraints). The first part, the recording/controlling of an external environment, seemed pretty sound. Then I remembered simulator systems. A computer-generated 3D model of a house, which is animated in real time so that the user can 'walk' around it, is neither recording nor controlling an external environment. Bother. For a moment, I thought we had the thing nailed.

If anybody out there thinks that he has a definition which is accurate and succinct, I should be pleased to hear from him. Meanwhile, having rubbished everybody else's definition, I'm certainly not about to attempt my own. Instead, here is my answer to 'How can you tell a real time system programmer?' Easy mate. They're the ones that aren't paid enough.

GREAT EMBEDDED SYSTEMS OF THE WORLD
The Docklands Light Railway

The small locomotive car, Number 4, came whining and buzzing up the gradient from Westferry with a full load of passengers. Four young children were sat at the front of the train, sometimes with noses pressed to the windscreen, mostly laughing and pointing to the track ahead. The train appeared round the corner, flanges grinding noisily on the tight curve, with loud threats of speed, but as it reached West India Quay... Hold the DH Lawrence, what do you mean the children were sat at the *front* of the train? What happened to the driver?

The Docklands Light Railway, opened by HM the Q in 1987, with 12.1 km of track in a 'Y' shape, running from the City of London in the west and Stratford in the east down to the tip of the Isle of Dogs; boasting 15 stations (14 of which were purpose-built), 11 electrically-powered passenger trains, each consisting of a single 90 ft car with an articulated joint in the middle; and more brick viaducts, cuttings, embankments and elevated concrete sections than you could find in five years' worth of Hornby catalogues.

But to return to the trains themselves. Beneath the smart red, white and blue livery, each car boasts an ability previously only vouchsafed to Thomas the Tank Engine and his friends: the ability to think for itself. A host of assorted on-board microprocessors collaborate to drive the train from station to station, accelerating away smoothly, slowing on corners and speeding up for the straight bits, going faster if the train is behind schedule, pulling into the station, halting accurately alongside the platform so that passengers can get out - a skill that sometimes eludes human drivers on the London Underground - and unlocking the doors. Other (independent) systems watch suspiciously, ready to hit the brakes if their silicon colleagues drive the train too fast, or place themselves on a collision course.

Independent systems watch suspiciously, ready to hit the brakes if their silicon colleagues drive the train too fast.

Technophobes will be pleased to hear that the trains are not entirely automatic. Each vehicle carries a DLR employee, grandly titled the 'train captain', who closes the doors and authorises the train to proceed to the next station. He (or she) accomplishes this using a key-switch control set into the lintel of the doors, duplicated at each of eight positions so that he may wander freely around the car. Between stops, he checks the tickets.

Smart trains

Figure 1 shows a schematic diagram of the various components of the automatic train control. The ATO (Automatic Train Operation) system is what Tomorrow's World might term 'the train's brain'. This is a Motorola 68000 processor based system, mounted in a VME bus rack together with various intelligent I/O boards. The processor runs the pSOS real time operating system, using application software written in C. It is this software that is responsible for driving the train from one station to the next. The sequence of events is as follows. At each station, the ATO establishes contact with a central computing system via a docking data link (DDL) - a box placed on the sleepers in between the rails. Information transmitted to/from the DDL to the vehicle lets the scheduling computer keep track of all the trains in the system.

When the train is safely 'docked' at a station - ie is communicating successfully with the DDL - the ATO unlocks the doors on the platform side. A few moments before the train is scheduled to leave, a tone is sounded over the PA system, to remind the

train captain to close the doors.

Once the train has left the station, the ATO is on its own. It has stored in ROM a set of 'run profiles', which provide instructions for negotiating the section of track between any pair of adjacent stations: 'Accelerate to reach 60 km/h in 200 metres, hold speed for 400 metres, apply brake to reach crawling speed within 50 metres, crawl until DDL link is reached, then halt'. Data from the vehicle's axles enables the ATO to determine the current speed and distance travelled. Two run profiles are held for each track section. One is for minimum journey time, and is used only when the train is running late. The other, which compared with the first saves about 30% of the energy at a cost of a 10% increase in journey time, is an energy saving profile which allows the car to coast whenever possible.

Not all the scheduling consists of straightforward station-to-station runs. The most striking example of a complication, which will be familiar to those of you who have ridden on the DLR, is the Canary Wharf stop. A glance at the tube map in the front of your diary will show you that, if you think of the DLR layout as a 'Y', then Canary Wharf station is situated on the upright, one station from the junction. In order to assign priorities to northbound trains crossing the junction (known as the Delta Junction), the scheduling computer at Poplar - actually two DEC minis running in synch, one providing 'hot backup' for the other - expects all trains to stop at Canary Wharf. There is just one teeny problem with this: when the DLR was opened, and despite the fact that it appeared on all the maps of the railway, Canary Wharf station had not been built. Trains stopped for quarter of a minute in the middle of an elevated section, with nothing either side of the track except a 40 foot drop to the concrete of the dock. Since the station was going to be built in any event, and there were more pressing matters requiring money spent than aesthetic alterations to the software, it was decided to let trains continue to pause at that bleak spot. They still do - but the construction of Canary Wharf station now proceeds apace.

Safety measures

The ATO system drives the train, but the ATP (Automatic Train Protection) system looks after the passengers' best interests. It has two responsibilities: to enforce speed limits and to prevent 'unsafe train movements' - eg two trains entering the same track section. The ATP determines the speed of the vehicle by means of cables laid across the track at intervals. The gaps between

Trains stopped for fifteen seconds on an elevated section with nothing on either side except a 40ft drop.

the cables are contrived so that a train travelling at the maximum permitted speed will cross a cable once a second. A signal transmitted by these cables is picked up by a coil mounted on the centre bogie as it passes overhead, and an 8749 microprocessor based system is informed of the transition. If another cable transposition is detected within one second then the 8749 will apply the brakes. Naturally, this is something that should never happen if the ATO is functioning correctly. (The speed monitor system is actually based on dual 8749 processors. This is a safety critical feature. The two processors execute similar code, duplicating calculations and comparing results. If any discrepancy is found between these results, the emergency brake is applied. This design principal is used in other areas; for example, the DDL contains dual 68008 processors.)

The other side of the ATP's duties - the prevention of unsafe train movements - is accomplished with a modified version of a standard BR signalling system based on Solid State Interlockings. These establish paths for the trains from one signal to the next. Audio frequency reed track circuits detect the presence of trains and transmit the data to a central group of SSIs. (The SSIs contain three processors which triplicate calculations, but require only two in agreement to function. This allows the high safety standard to be maintained, while supplying a degree of redundancy necessary for practical operation of the railway). The transfer of information to the trains differs greatly from the conventional system. Excepting those areas of track not under computer control - for example at the Operation and Maintenance Centre where the trains are 'stabled' (correct jargon) - there are no coloured light signals. Instead of getting the green light to move forward, DLR vehicles detect a 408 Hz signal, interpreted as 'permission to proceed', which is transmitted along the rails and picked up by two coils mounted ahead of the leading axle. If, for any reason, the signal should stop, then the ATP ensures that the train stops too.

The Developers

The DLR's automatic control system was designed and built by GEC General Signals at the company's Borehamwood and Manchester sites. Although the software component part of the project naturally employed systematic development techniques (such as source code configuration control), there was no use of CASE, or mathematical proving of systems. GEC had a very rigorous approach for the validation of safety-critical code, which used design-independent personnel to inspect code *after* it had been finished and tested. Since the work was done some years ago, the Ada language was not an option. It is interesting to note the use of C on this project. In the writer's opinion, it is this type of work, rather than its frequent and voguish use for MS-DOS applications, which best exploits C's high level/low level features. Cross-development for the embedded microprocessors was mostly done using HP host systems and development tools. The company is now extending the systems for the DLR extensions. These include an extra track section, running underground into the City of London, and new two-car trains, which will increase the capacity of the railway.

Finally, please spare a thought for the DLR's train captains, whose job (and, if they are human, surely their secret hope) is to drive the train in the event of ATO failure. Conventional trains have a 'dead man's handle', allowing automatic systems to take over if the driver is taken ill. On the DLR, they put in a human to step in and snatch the controls in case the train faints.

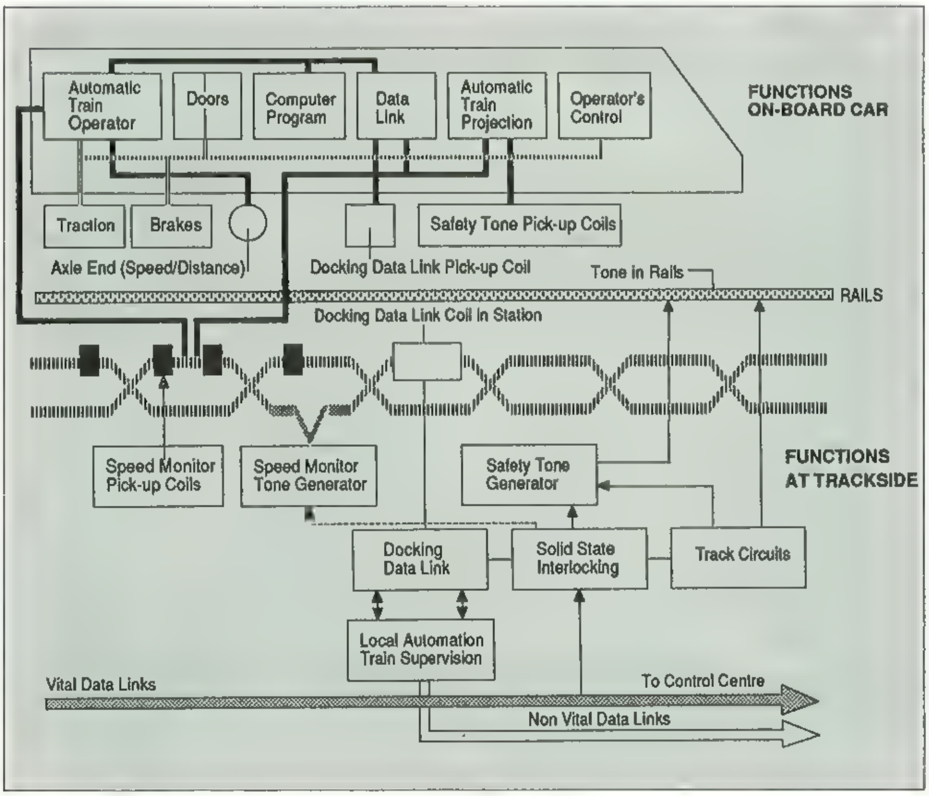


Figure 1 - Schematic diagram, showing the inter-related functions of automatic train control

Compiling at a Moving Target

The classical design of cross-compiler, which was the method used to create the original generation of Hewlett Packard tools for 8-bit microprocessors (8085/Z80, 6800, 6809), incorporates a code generator which must be rewritten for each target processor. The problems with this were a) that it results in a large volume of unique program code, with correspondingly high defect rates and b) excessive time is spent designing and writing a new code generator for each processor. To address these difficulties, we developed a new technology that increased reuse of code and separated the burden of compiler writing from that of describing the target processor's instruction set. The compiler that we produced with this technology - the AxLS C compiler - has been benchmarked against the industry's leading products: its performance is excellent. This article provides an overview of our compiler technology and takes a closer look at one particular aspect of code generation: the costing of statement trees.

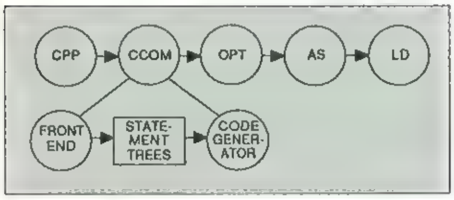


Figure 1 Language tool processes with exploded view of ccom.

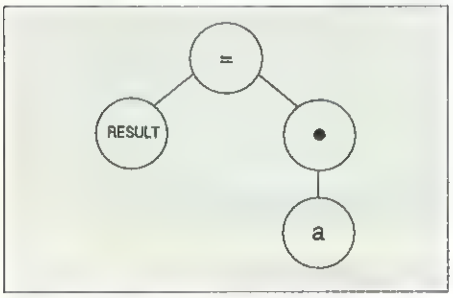


Figure 2 - Example program tree for 'result = a'

Authors Chris Jones (co-designer of AxLS compiler) and Chris Tilt (Software Engineer) work at Hewlett-Packard's Logic Systems Division in Colorado Springs, USA.

The AxLS C compiler range, described in this article, is available for 68000, 68020, 68030, 80186 and 80188 target processors, with plans for the 68040 and 80386. More information from Hewlett-Packard on 0734 777838.

This article is an extract from a more detailed paper. To obtain a copy of the original, please call Ray Daniel at HP on 0344 773100.

Introduction to AxLS

The complete set of Advanced Cross Language System (AxLS) compiler tools includes the C preprocessor, C compiler, optimiser, assembler and the linker (see Figure 1). We will concentrate on the constituent parts of the `ccom` program (the C compiler), which consists of a front end, an intermediate program tree and a code generator.

The `ccom` program is built from a common set of sources for all target processors. We maximise the advantage of code reuse by employing a common front end and code generator.

The front end is that part of a compiler which recognises the syntax and semantics of a language. It has the responsibility of parsing the C source code, detecting and reporting errors, performing high level optimisations and generating statement trees to be passed to the code generator. The front end receives input from the C preprocessor and parses it in a single pass using a yacc generated parser. It produces an internal data structure, representing the original source. A second pass is made over this structure, generating numerous statement trees (one for each statement in the original program). Finally, the code generator is called; it is invoked once for each statement tree. To support a new target processor, the only change to `ccom` is the definition of the processor in Template Description Language (TDL). The TDL source is compiled into static data and can then be linked to the standard front end and the code generator object files.

Statement Trees

The statement trees output by the front end specify the individual operations that make up a statement, expression or definition in the C program. The collection of all of the trees represents the operations of the original source program. Figure 2 shows an example of a tree for the statement `result = a`.

Each node in the tree is a low-level operator with defined semantics. For example, assuming static 32-bit integers `result` and `a`, the operator represented by the symbol = above is **StatAssignVoidPtrAny**, with a size attribute value 4 (bytes). The operator is defined as having a pointer as its left child and a 4-byte right child. Its purpose is to generate code to assign the right child's value to the address pointed to by the left child. Similarly, the operator represented by the symbol • is **PointerDereferenceAnyPtr**, with size attribute 4 and one pointer child. Its purpose is to generate code to pass up 4 bytes at the location pointed to by its child.

Finally, the nodes labelled `result` and `a` in Figure 2 both represent the **StaticReference** operator, with a name attribute of `result` and `a` respectively. **StaticReference** is a leaf (no children) operator. It generates the code to produce a pointer to the memory location allocated for the named static data item.

Code Generation

The code generator portion of `ccom` converts these statement trees into output assembly language, using processor-specific information held in tables. It accomplishes this by selecting templates to 'cover' the program tree. A template contains the assembly language code required to perform the operations in the subtree it covers. For example, a template for the 68000 that covers the tree in the last example might generate the code:

```
MOVE.L a, result
```

The code generator first selects the templates it will use to cover the statement tree in a cost effective manner. It then allocates registers to hold temporary results (usually termed **tns**, for temporary **n**ames) and generates the assembly language code.

Let's look at the code generator mechanism in more detail. (The example that follows has been simplified for tutorial purposes.) The first pass made over the statement tree determines the cost of all ways to cover the tree with templates. This pass is made from the bottom of the tree upwards, and determines which templates cover the subtree rooted at the current node. Consider the example in Figure 2. The first node to be visited is `result`. Figure 3 illustrates this.

Template #1 is the first template that matches this subtree. It covers the StaticReference operator whose **name** attribute is *result*. The cost of Template #1 is 6 bytes and 4 machines cycles. As-

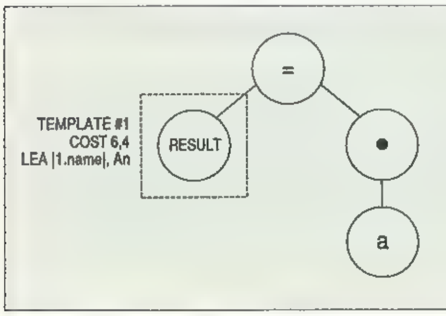


Figure 3

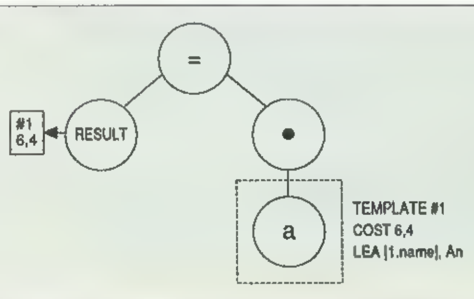


Figure 4

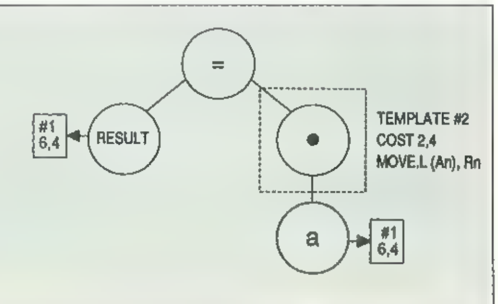


Figure 5

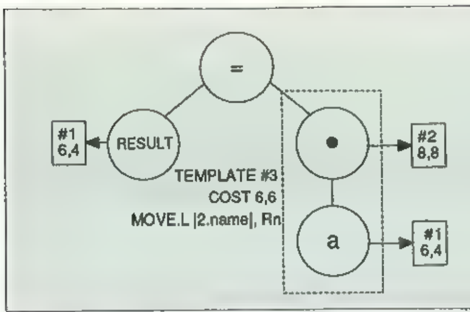


Figure 6

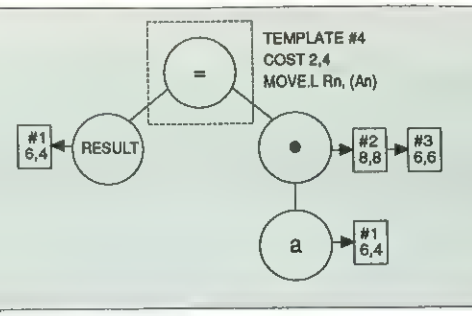


Figure 7

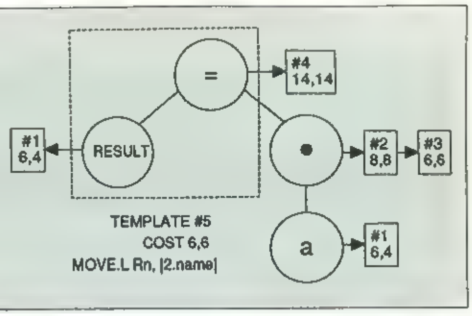


Figure 8

suming that A0 were selected as the result register, then the code generated by Template #1 would be:

```
LEA result, A0
```

The next node visited in a bottom-up tree walk is labelled a. Once again, Template #1 matches the **StaticReference** operator resulting in the situation shown in Figure 4. Note also that the match of Template #1 at the node labelled `result` has been reduced to a 'decoration' - sorry about the seasonal terminology - on the statement tree. At Figure 5 a new template, Template #2, has been found to match the **PointerDereferenceAnyPtr** operator labelled •. The **size** attribute value of 4 means that four bytes are to be read from the child location. Template #2's cost is 6 bytes and 4 machines cycles. Here is the code that it would generate, if A0 and D0 were selected as its input register (or '**use tn**') and result register ('**eval tn**') respectively:

```
MOVE.L (A0), D0
```

As expected, the match of the node labelled a has been reduced to a decoration.

The next template match that occurs, shown in Figure 6, is the first for a template covering more than one operator. Template #3 covers both the PointerDereferenceAnyPtr operator and the **StaticReference** operator and, if D0 was chosen as its **eval tn**, would generate:

```
MOVE.L a, D0
```

at a cost of 6 bytes and 6 machines cycles. Now look at the decoration showing Template #2's match of the node labelled •. You will see that it has a cost of 8,8, ie the cost of Template #2 (2,4) plus the cost of Template #1 (6,4), which covers the node labelled a. This is clearly the cost of covering the entire subtree rooted at the node labelled •, but note that the logic is not completely straightforward. Also included in the cost of moving from Template #1's **eval tn** to Template #2's **use tn**. Both Template #2's **use tn** and Template #1's **eval tn** are 'An' (any address register). So the algorithm assumes that, since the two register sets are the same, the register allocator will choose the same register for both, resulting in a move cost of zero.

At the root

Finally, costing begins on the root node. Figures 7-11 show the successive application of different templates to the root node. Figure 7 shows the smallest template covering the root StAssignVoidPtrAny operator, whose **size** attribute has value 4 and is represented by the node labelled =. The cost of Template #4's decoration, which is shown in Figure 8, is

(14,14) = #4's (2,4) + #1's (6,4) + #3's (6,6).

Template #3's cost is used, since it is the cheapest covering of its subtree. As before, all move costs are assumed to be zero because all use tn and **eval tn** sets overlap. Templates #6 and #7 each cover three operators, and the last template, Template #8, covers the entire tree.

The real story

The example just discussed is inaccurate. In reality, the `tmaker` (template generation) program sorts the templates in order of decreasing complexity. So, in the above example, the matches that cover just one operator would be tried *after* the matches that cover multinode subtrees.

Look at the node labelled • in the example above. Template #3 would have been matched first. Template #2 would never have been recorded on the statement tree decorations, because it costs more than Template #3 and has the same **eval tn** set. Similarly, at the root, Template #7 (which covers the entire statement tree) would have been matched first. Subsequent matches of Templates #4, #5 and #6 would not have been recorded.

In fact, all the templates described above are artificial. This is because most 68000 instructions allow a variety of address modes. The job of elaborating all templates resulting from all address modes, across all operations, is enormous and impractical. To avoid such repetitions, address modes are expressed as a separate group of 'partial templates'.

Partial templates, like normal templates, cover one or more operators in the program tree. Unlike normal templates, the code associated with address modes includes no opcode - only the operand syntax required to use the address mode. For example, the 'absolute long' address mode on the 68000 (and most processors) is used to read from a memory location at an absolute address. This address mode would cover the subtree shown in Figure 12, and could have the code `2.name` (meaning 'print the second operator's name attribute', here 'staticvar'). This address mode could then be used in conjunction with any template whose root operator's 68000 instruction could use the absolute long address mode. For example, in the above `result = a` case, Template #7 is a combination of a template covering just the = node and the absolute long address mode.

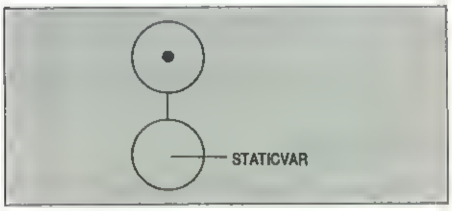


Figure 12

When the costing pass over the statement tree is complete, the task of selecting templates is a simple one. At the root node, the least expensive template is selected. Then, using a recursive algorithm, the root of each subtree not covered by the selected template is visited. From this is built the 'template tree' on which subsequent passes, such as the register allocation pass, will operate. Unfortunately, there is no room here to describe these passes; these belong, as Enid Blyton might put it, to another story.

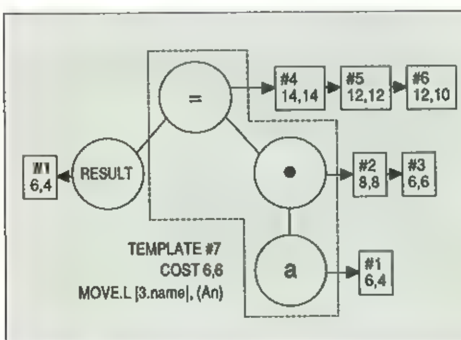


Figure 9

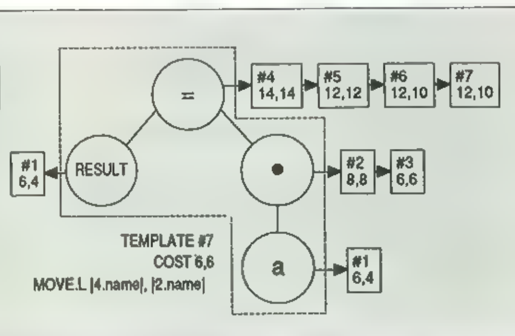


Figure 10

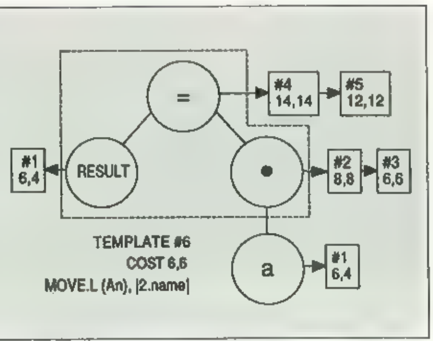


Figure 11

Operating Systems in Real Time

This begins like an extract from one of Datalink's 'The Project I'll Never Forget' articles, but improves. I used to work for a small company whose main product was an industrial weighing machine. Based on a Z80 processor, the machine could be programmed to do such things as print bar-coded labels, count the number of objects moved across its scales and so on. The machine's software, which we had to add to and modify for each customer, consisted of about 8 KB (object) of undocumented assembly language, written by a man who had long since departed. There was no formal operating system or kernel. The code consisted of an interrupt handler (to get in weigh values from an A to D) and an endless loop (in which you had to do everything else).

The software worked quite well, but was mined with traps for the unwary: obscure, but vital, system code which secretly relied upon data labels being declared in a certain order, a non re-entrant ASCII division routine that gave the right answer 96% of the time. When the applications got complex, the coding got tough. Because of the lack of structure in the base software, a lot of the code that we wrote was not reusable (something that was hard to explain to the electrical engineers around us). The backlog grew apace. To improve matters, and because I am a tiresome innovator, I implemented a C language interface to the system, quadrupling both my productivity and the size of EPROM required. My popularity went up (marginally) with the management and diminished considerably with the hardware department. The C language fix was a stopgap: what was needed was a complete redesign of the system software. Naturally, we did not have the resources to do this.

I had never heard of an off-the-shelf real time operating system. Even when I started collecting data for this survey, now steeped in journalistic experience (which is all about knowing products and very little dirtying of the hands with code), I naively thought that might be a dozen, packages in all. In fact, there are many more than the twenty-two systems listed here. If I have failed to give a complete picture (my apologies to those I have left out), then I hope that this piece will at least indicate the scope of what is available.

The packages listed below are not, by any means, all in direct competition with each other, for I have bundled together like and unlike. They range from kernel-only packages like pSOS+ to the vast, UNIX-based REAL/IX; there is the specialised transputer-only operating system Helios, and the tiny HI8 kernel, which is a partial implementation of the TRON project specification. (The TRON project is a Japanese idea, of Douglas Adams/science fiction proportions, for making all computers compatible with each other. The long-term aim is that your microprocessor-based door-closer, say, will be able to talk to your intelligent tea-maker; naturally your PC will be able to communicate with both of them).

The survey questions are largely self-explanatory. The time taken for a context switch is an indication of how much of *your* processing time will be lost to the operating system's housekeeping activities. Interrupt latency, for those who have not previously encountered the jargon, is the length of the time interval between an interrupt occurring and the designated handler routine gaining control. This is important if, to borrow an example from an RMX promotion, you are writing software for an interrupt-driven anti-lock braking system. The quoted prices are only intended to give you a general idea of how much a system might cost; there was no space to list details of all the configurations available. Of all the manufacturers and distributors that I contacted, only Digital Research felt unable to give me sample prices. Why the coyness, DR? Surely Flexos does not cost *that* much?

Finally, a reminder that, since all these figures are supplied by their manufacturers, they should be taken with a little added seasoning here and there. And Terry: if you're out there somewhere, please note that there's a few OSs that support the Z80/64180. It's got to be better than spending the rest of your days rewriting `WEIGHX`.

Twenty-Two Real Time Operating Systems

C EXECUTIVE by JMI Software Consultants.

Target processors: ▲ 8080/8085, 8088-80386, 6809, 68000-68030, Series 32000, Z80, Hitachi 64180, Intergraph Clipper, TI 34010, AMD 29000, Sun SPARC. Various RISC processors under development
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Re-entrant support: yes; position independent code support: no
Memory requirements: ▲ Kernel 6 KB - 30 KB, File system - 10 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 87 µs (68000 at 10 MHz)
Memory protection/MMU use▲ Memory space protected, no MMU support
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ (No figures provided)
Inter-task communication facilities: ▲ Public data queues, systems events, semaphores, files
Standard I/O drivers: ▲ UART, floppy disk, hard disk
Networking interfaces: ▲ None
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ Yes, can exchange files with MS-DOS
User interface: ▲ No
Multi-user: ▲ Yes
High level languages: ▲ C (Various vendors)
Choice development language: ▲ C
Choice development method: ▲ Target using CE-VIEW
Price of single user development system, price per run-time (100 off): ▲ £600, £60
Nearest perceived competitor: ▲ VRTX
Example applications: ▲ Factory automation communication controller, point of sale cash registers, radar display
Age of product, current version: ▲ 8 years, V2.3
Run-time copies and development systems in use: ▲ 100,000, 115
Distributor contact: ▲ Paul Evans, Real Time Systems, P O Box 70, Viking House, Nelson Street, Douglas, Isle of Man. Tel: 0624 661400

Notes: ▲ *Kernel written in ANSI C provides good portability. Device independent I/O.*

Flexos by Digital Research

Target processors: ▲ 80186-80386, NEC V series, 68000
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 80-180 KB, file system 150 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 50-60 µs (80386 at 24 MHz)
Memory protection/MMU use▲ Uses MMU where available
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Available through drivers
Interrupt latency: ▲ 18 µs (80386 at 24 MHz)
Inter-task communication facilities: ▲ Pipes
Standard I/O drivers: ▲ Floppy disk, serial port, console, hard disk, parallel port
Networking interfaces: ▲ IBM token ring, TCP/IP, ARCNet, Ethernet
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ MS-DOS 4.X compatible
User interface: ▲ MS-DOS compatible
Multi-user: ▲ Yes
High level languages: ▲ C, Pascal, FORTRAN 77, COBOL
Choice development language: ▲ C (Metaware)
Choice development method: ▲ Directly on target
Price of single user development system, price per run-time (100 off): ▲ Refused to supply
Nearest perceived competitor: ▲ RMX family
Example applications: ▲ Various EPOS terminals
Age of product, current version: ▲ 4 years, V2.11
Run-time copies and development systems in use: ▲ 200,000, 350
Distributor contact: ▲ Phil Perkins, Digital Research, Oxford House, Oxford Street, Newbury, Berkshire. Tel: 0635 35304

Notes: ▲ *Supports various add-ons, notably X/GEM, a multi-tasking graphics environment, based on the GEM interface*

Helios by Perihelion Software Ltd

Target processors: ▲ Inmos T800, T414 T425 transputers
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Nucleus 64 KB, Nucleus data 64 KB, Shell 150 KB, POSIX library 15 KB, C library 25 KB, maths 15 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ <1 µs (T800 at 20 MHz, done in hardware)
Memory protection/MMU use▲ No
Prioritised scheduling: ▲ No
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 1 µs (T800 at 20 MHz)
Inter-task communication facilities: ▲ POSIX standard pipes, message passing
Standard I/O drivers: ▲ Centronics, RS232, floppy disk, hard disk, mouse, XWindows
Networking interfaces: ▲ Transputer links, Ethernet
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ MS-DOS
User interface: ▲ UNIX C-Shell
Multi-user: ▲ Yes
High level languages: ▲ C (in house), FORTRAN, Pascal, Modula-2, BASIC, Ada, occam
Choice development language: ▲ C
Choice development method: ▲ Directly on target
Price of single user development system, price per run-time (100 off): ▲ £750, £125
Nearest perceived competitor: ▲ Claimed none; 'Helios is fairly unique'
Example applications: ▲ Super computer CCT board

layout, graphics engine, general purpose computer
Age of product, current version: ▲ 18 months, V1.1
Run-time copies and development systems in use: ▲ Less than 1000
Distributor contact: ▲ Distributed Software Ltd, 670 Aztec West, Bristol BS12 45D. Tel 0454 612777

Notes: ▲ *Specialised support for transputer in UNIX-like environment. Distributed multi-processor OS, distribution mechanism transparent to user.*

HI8 by Hitachi

Target processors: ▲ H8/532 microcontroller
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 1.9 KB - 4.7 KB, Debugger 2.5 KB - 5 KB, Data 18 bytes (!) min
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ Not known
Maximum no of tasks: ▲ 64
Memory protection/MMU use▲ No
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Not known
Interrupt handler interface: ▲ Not known
Interrupt latency: ▲ 15 µs (H8 at 10 MHz)
Inter-task communication facilities: ▲ Mailbox, semaphores, flags
Standard I/O drivers: ▲ Printer driver, console driver, timer handler
Networking interfaces: ▲ None
Standard real time s/w interfaces: ▲ Conforms to level 3 of ITRON specification
Disk file system: ▲ No
User interface: ▲ N/A
Multi-user: ▲ No
High level languages: ▲ C
Choice development language: ▲ C/Assembler
Choice development method: ▲ HIDE (see note)
Price of single user development system, run-time for 100 copies: ▲ Not yet finalised
Nearest perceived competitor: ▲ VRTX
Example applications: ▲ Engine control
Age of product, current version▲ First release December 1989, V1.0
Run-time copies and development systems in use: ▲ N/A
Distributor contact: ▲ Steve Evans/Dave Wallace, Hitachi Europe Limited, 21 Upton Road, Watford, Herts. Tel: 0923 246488

Notes: ▲ *Hitachi Integrated Development Environment (HIDE) runs on the PC in conjunction with PC plug-in H8/532 Evaluation Boards. This provides the user with a fully integrated, PC-based development environment. Software adheres to the ITRON specification, which is part of the ambitious TRON project*

MIRAGE by Swift Computer Systems Ltd

Target processors: ▲ 68000-68030
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 36 KB, Files 15 KB, CLI 28 KB, Maths 16 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 80 µs (68000 at 12 MHz)
Memory protection/MMU use▲ No
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 42.3 µs (68000 at 12 MHz)
Inter-task communication facilities: ▲ Semaphores, events, locks, rendezvous
Standard I/O drivers: ▲ Terminals, serial, parallel, floppy and hard disks, tape drives etc
Networking interfaces: ▲ Ethernet, Midi, IEEE-488, TCP/IP
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ Yes, access to MS-DOS and pSystem (UCSD)
User interface: ▲ Proprietary
Multi-user: ▲ Yes
High level languages: ▲ Pascal, FORTRAN-77, BASIC (all in-house); APL, FORTH, PROLOG
Choice development language: ▲ None
Choice development method: ▲ Directly on Target Hardware
Price of single user development system, price per run-time (100 off): ▲ £450, £57.50
Nearest perceived competitor: ▲ UNIX
Example applications: ▲ Vibration analyser used to test vehicle stress points, workstation used to train air traffic controllers, brewery production line control
Age of product, current version: ▲ 10 years, V2.0.12
Run-time copies and development systems in use: ▲ 20,000, 4000
Distributor contact: ▲ William Dowling. Sahara Software, South Bank Technopark, 90 London Road, SE1 6LN. Tel: 01 922 8850

Notes: ▲ *Developed in UK. Written in assembly language so is fast. Wide range of high-level languages supported.*

MTOS by Industrial Programming Inc

Target processors: ▲ 8086-80486, 68000-68030, 88000, National 32000 series
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel about 24 KB, data 2 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 38 µs (80386 at 20 MHz)
Memory protection/MMU use▲ 80386 uses full protected mode
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 5 µs (68020 at 20 MHz)
Inter-task communication facilities: ▲ Event flags, semaphores, signals
Standard I/O drivers: ▲ Various
Networking interfaces: ▲ UDP, TFTP. TCP/IP under development
Standard real time s/w interfaces: ▲ Under development
Disk file system: ▲ MS-DOS compatible
User interface: ▲ N/A, Kernel only. Driver to allow tasks to access most UNIX services
Multi-user: ▲ N/A
High level languages: ▲ C (Various vendors including GreenHills, Oregon Software, Microsoft and Metaware), Ada, Pascal
Choice development language: ▲ C
Choice development method: ▲ Any MTOS systems, code is portable
Price of single user development system, price per run-time (100 off): ▲ £3950, £165
Nearest perceived competitors: ▲ pSOS, VRTX, VxWORKS
Example applications: ▲ AWACS radar system, process control systems. IBM 5080 graphics workstations
Age of product, current version▲ 10 years, version varies with target
Run-time copies and development systems in use: ▲ 100,000, 2000
Distributor contact: ▲ David Bullimore, Real Time Products, Canterbury House, Birmingham B3 1LH. Tel: 021 236 8070

Notes: ▲ *Application code portable between targets. True multiprocessing with dynamic CPU balancing (so can be used in fault tolerant systems). PC/AT version available.*

OS-9/68k by Microware Systems Corporation

Target processors: ▲ 680x0 family
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 26 KB, Disk file manager 9 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 57 µs (68020 at 20 MHz)
Memory protection/MMU use▲ Yes, using optional system security module
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes, using FS1RQ system call
Interrupt latency: ▲ 11.1 µs (68020 at 20 MHz)
Inter-task communication facilities: ▲ Pipes, data modules, signals, events
Standard I/O drivers: ▲ Character (serial/parallel), Disk (floppy/hard), Tape
Networking interfaces: ▲ OS-9/NET - Arcnet, Serial, Shared RAM, Ethernet; TCP/IP - Ethernet
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ Yes, with option to access MS-DOS files
User interface: ▲ Yes, UNIX-like shell
Multi-user: ▲ Yes
High level languages: ▲ C, FORTRAN, Pascal, BASIC (in-house)
Choice development language: ▲ C
Choice development method: ▲ On target, directly
Price of single user development system, price per run-time (100 off): ▲ £200, £50
Nearest perceived competitor: ▲ UNIX
Example applications: ▲ Power distribution, assembly line robotics, interactive compact disks (CD-I), image processing
Age of product, current version: ▲ 7 years, V2.3 (OS-9/68k)
Approx run-time copies and development systems in use: ▲ 250,000, 5000
Distributor contact: ▲ M Quelch, Microware Systems (UK) Ltd, 1626 Parkway, Solent Business Park, Fareham, Hants PO15 7AH. Tel: 0489 886699

Notes: ▲ *Wide range of peripheral drivers available. Easy scalability of OS to suit system application. Enhanced version of OS-9, called OS-9000 has been announced, adding support for 80386 and RISC processors*

PDOS by Eyring Researching Inc

Target processors: ▲ 68xxx family, 88000
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 4 KB, Files 10 KB, data 4 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 25 µs (68020 at 25 MHz)
Maximum no of tasks: ▲ 127
Memory protection/MMU use▲ No, under development for 68030
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Standard primitive installs custom handler can be loaded by task
Interrupt latency: ▲ Estimated 2 µs max (68020 at 25 MHz)
Inter-task communication facilities: ▲ Messages
Standard I/O drivers: ▲ Serial, disk, parallel, graphics
Networking interfaces: ▲ Ethernet (EMOD).
Standard real time s/w interfaces: ▲ No, ORKID and 88-open drivers under development
Disk file system: ▲ Yes, to own tandard. MS-DOS and UNIX interchange under development
User interface: ▲ Yes, own standard. POSIX version under development
Multi-user: ▲ Yes, not as special function
High level languages: ▲ Pascal, BASIC (both in-house) C, FORTRAN-77
Choice development language: ▲ None
Choice development method: ▲ On target
Price of single user development system, price per run-time (100 off): ▲ £850, £25
Nearest perceived competitor: ▲ OS-9, VRTX
Example applications: ▲ NASA platforms, process control
Age of product, current version: ▲ 15 years, V3.3e
Run-time copies and development systems in use: ▲ Unknown, 'a lot'

Distributor contact: ▲ Eyrisoft Limited, Etwall St, Derby DE3 3DT. Tel: 0332 384978

Notes: ▲ *Manufacturer claims that PDOS is rated in the top three for real time response.*

PromDOS by Hexatron

Target processors: ▲ 8088 - 80386, NEC V20, V25, V40
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Total system 32 KB ROM, 4 KB RAM
Pre-emptive multi-tasking: ▲ Optional
Context switch: ▲ 100 µs (V25 at 8 MHz)
Maximum no of tasks: ▲ 16
Memory protection/MMU use▲ No
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 80 µs (V25 at 8 MHz)
Inter-task communication facilities: ▲ Global Message area
Standard I/O drivers: ▲ All MS-DOS type drivers
Networking interfaces: ▲ None
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ MS-DOS compatible
User interface: ▲ MS-DOS, debugger, Command line BASIC
Multi-user: ▲ No
High level languages: ▲ All PC compilers
Choice development language: ▲ None
Choice development method: ▲ Direct on target
Price of single user development system, price per run-time (100 off): ▲ £995, £25
Nearest perceived competitor: ▲ MS-DOS, CDOS (for users who don't understand real time), RMX (for those who do)
Example applications: ▲ TV control on M25, industrial terminals, telephone monitor
Age of product, current version▲ 3 years, V3.2
Run-time copies and development systems in use: ▲ 3000, 200
Distributor contact: ▲ Andy Butler, Hexatron, Unit 5, Business Centre, Avenue One, Letchworth, Herts. Tel: 0462 675530

Notes: ▲ *MS-DOS compatibility combined with real time OS, optional multi-tasking supports programs developed with standard MS-DOS tools*

pSOS+ by Software Components Group

Target processors: ▲ 68xxx, 8086-80386, AMD 29000, 88000
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 12 KB, File system 21 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 19 µs (68020 at 25 MHz)
Memory protection/MMU use▲ MMU support on 68030 and 88000 versions
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 7 µs (68020 at 25 MHz)
Inter-task communication facilities: ▲ Queues, events semaphores
Standard I/O drivers: ▲ UARTs, Clocks, SCSI, VME
Networking interfaces: ▲ Ethernet, TCP/IP
Standard real time s/w interfaces: ▲ RTEID/ORKID
Disk file system: ▲ Proprietary and MS-DOS formats
User interface: ▲ No, kernel only
Multi-user: ▲ N/A
High level languages: ▲ Telesoft Ada, C, Pascal, FORTRAN
Choice development language: ▲ C
Choice development method: ▲ Host/target linked system with XRAY source level debugger
Price of single user development system, price per run-time (100 off): ▲ £2900, £50
Nearest perceived competitor: ▲ VRTX
Example applications: ▲ Emulators, logic analysers, process control
Age of product, current version: ▲ 7 years, V4.6
Run-time copies and development systems in use: ▲ Unknown, 'a lot'
Distributor contact: ▲ Geoff Revill, Thame Microsystems Limited, Thame Park Road, Oxfordshire OX9 3UQ. Tel: 0841 261486

Notes: ▲ *First with 88000 support. Claimed to be only kernel with true source level multi-tasking debugger. Transparent multiprocessing. Established record of reliability*

QNX by Quantum Software Systems

Target processors: ▲ 8088-80386
ROMable: ▲ Impractical
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 115 KB. Requires 256 KB for simple applications, up to 4 MB of RAM for UNIX style applications
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 158 µs (80286 at 12 MHz)
Maximum no of tasks: ▲ 150
Memory protection/MMU use▲ Yes, uses protected mode where available
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes, as privileged tasks
Interrupt latency: ▲ Unknown. Handler is entered from hardware vectoring
Inter-task communication facilities: ▲ Messages system
Standard I/O drivers: ▲ Hard/floppy disks (ST506, SCSI, ESDI), Keyboard, screen, Serial (terminal and modem-management), Parallel
Networking interfaces: ▲ TCP/IP (Genus-MARI, CMC); X.25 (Corman)
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ Proprietary, can access MS-DOS
User interface: ▲ Yes. UNIX-like shell
Multi-user: ▲ Yes
High level languages: ▲ C, BASIC (in-house), FORTRAN 77, Pascal (under development)
Choice development language: ▲ C
Choice development method: ▲ On target
Price of single user development system, price per run-time (100 off): ▲ £350, £135
Nearest perceived competitor: ▲ OS-9
Example applications: ▲ Air Traffic Control simulator, voicing systems, disk error surface testing
Age of product, current version▲ 7 years, V2.15C (3.15C is network version)
Run-time copies and development systems in use: ▲ 85,000, 4000
Distributor contact: ▲ Chris Whitfield/Catherine Fyall, Genus Systems, 9A St Colme Street, Edinburgh. Tel: 031 225 6934

Notes: ▲ *Unusual 'virtual circuits' architecture allows reuse of code modules without re-linking. Exploits protected mode of 80286 processor.*

REAL/IX by Modcomp

Target processors: ▲ 68000-68030
ROMable: ▲ No
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Not known
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 30 µs (68030 at 33 MHz)
Maximum no of tasks: ▲ 256
Memory protection/MMU use▲ Yes
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ Not known
Inter-task communication facilities: ▲ Standard UNIX facilities
Standard I/O drivers: ▲ Not known
Networking interfaces: ▲ Not known
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ UNIX compatible
User interface: ▲ UNIX shell
Multi-user: ▲ Yes
High level languages: ▲ C, Pascal, FORTRAN, COBOL, Ada
Choice development language: ▲ C
Choice development method: ▲ On Modcomp 9730/9740 machine
Price of single user development system: ▲ £26,000 (including cost of 9720 machine)
Nearest perceived competitor: ▲ DEC VMS
Example applications: ▲ Industrial process control communications
Age of product, current version: ▲ 1 year, N/A
Run-time copies and development systems in use: ▲ Not known
Distributor contact: ▲ Modcomp, Molly Millars Lane, Wokingham. Tel: 0734 786808

Notes: ▲ *Real time version of UNIX, fully compliant with AT&T SVID and SVVS. Based on AT&T UNIX system V3 kernel.*

RMX I by Intel

Target processors: ▲ 8086-80486
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 24 KB , BIOS 64-100 KB, EIOS 13 KB, full size (with I/O human interface networking) 300 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 690 µs (8086 at 8 MHz)
Memory protection/MMU use▲ No MMU use, software mechanism protects OS data
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 60 µs (8086 at 8 MHz)
Inter-task communication facilities: ▲ Mailbox, Semaphore, Region, Multibus II, Message passing
Standard I/O drivers: ▲ USART, floppy, hard disk
Networking interfaces: ▲ RMXNET, X.25 INA960
Standard real time s/w interfaces: ▲ RMX
Disk file system: ▲ Proprietary, but can access MS-DOS files
User interface: ▲ Proprietary, some UNIX-like features
Multi-user: ▲ Yes
High level languages: ▲ Intel OMF: PL/M, C, Pascal, FORTRAN
Choice development language: ▲ PL/M
Choice development method: ▲ On target, with ICE and DOS hosted development debug environment
Price of single user development system, price per run-time (100 off): ▲ $3490, $76
Nearest perceived competitor: ▲ Not saying
Example applications: ▲ Railway power monitoring and control
Age of product, current version: ▲ 7 years, RMX I.8
Run-time copies and development systems in use: ▲ Not known
Distributor contact: ▲ Intel - Sales & Applications, Pipers Way, Swindon, Wiltshire FN3 1RJ. Tel: 0793 696000

Notes: ▲ *Faster response times than RMX II, because MMU slows processing, so it is possible to develop under RMX II and port debugged application down. Intel also supplies training for designing RMX and RMK systems, including device driver configurations - contact Susan Draper at Intel Customer Training for more information (0793 696539).*

RMX II by Intel

Target processors: ▲ 80286-80486
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ 40 KB min to 380 KB max (for full OS with I/O system, CLI and network)
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 123 µs (typical, 80386 at 20 MHz)

Memory protection/MMU use: ▲ Yes, 286 protected mode + OS object protection
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 60 µs (max, 80386 at 20 MHz)
Inter-task communication facilities: ▲ Mailbox, Semaphore, Region, Multibus II
Standard I/O drivers: ▲ Yes, USART, terminal drivers, graphics, floppy, tape, mini
Networking interfaces: ▲ RMXNET, (OSI) X.25 INA960
Standard real time s/w interfaces: ▲ Yes, Intel
Disk file system: ▲ Proprietary, but can access MS-DOS files
User interface: ▲ Proprietary, some UNIX like features and source control
Multi-user: ▲ Yes
High level languages: ▲ PL/M, C, Pascal, FORTRAN
Choice development language: ▲ PL/M
Choice development method: ▲ Either on target or MS-DOS hosted
Price of single user development system, price per run-time (100 off): ▲ $5500, $251
Nearest perceived competitor: ▲ VRTX
Example applications: ▲ Body scanner, stockbroking dealing systems
Current version▲ RMX II.4
Run-time copies and development systems in use: ▲ Not known
Distributor contact: ▲ See RMX I

RMX III by Intel

Target processors: ▲ 80386-80486, 386SX, 376
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ 56 KB min to 550 KB max (with full CLI and network interface)
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ Not known
Memory protection/MMU use▲ Yes, (386 native mode)
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ Not known
Inter-task communication facilities: ▲ Mailbox, Messages, Semaphore, Region, Multibus II
Standard I/O drivers: ▲ USART, terminals, graphics, floppy, tape, hard disks
Networking interfaces: ▲ RMXNET, (OSI) X.25 INA960
Standard real time s/w interfaces: ▲ RMX
Disk file system: ▲ Proprietary, can access MS-DOS files
User interface: ▲ Proprietary CLI with some UNIX features
Multi-user: ▲ Yes
High level languages: ▲ PL/M, C, FORTRAN, (OMF 286, 386)
Choice development language: ▲ PL/M
Choice development method: ▲ On target with MS-DOS host
Price of single user development system, price per run-time (100 off): ▲ $6500, $289
Nearest perceived competitor: ▲ VRTX 32
Example applications: ▲ Automated PCB inspection test
Age of product, current version▲ 6 months, RMX III.1
Run-time copies and development systems in use: ▲ Product released December 1989
Distributor contact: ▲ See RMX I

Notes: ▲ *32 bit OS, compatible with RMX II (supports 16 bit and 32 bit tasks simultaneously). Intended for maths - intensive applicators*

RMK by Intel

Target processors: ▲ 80386-80486, 80960
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ 8 KB - 27 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 20 µs (processor and speed anonymous!)
Memory protection/MMU use▲ Yes, 386 protected mode
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ Not known
Inter-task communication facilities: ▲ Mailbox, Messages, Semaphore, Multibus II
Standard I/O drivers: ▲ None (Kernel only)
Networking interfaces: ▲ None
Disk file system: ▲ None
User interface: ▲ None
High level languages: ▲ Intel OMF 386, compilers generating COFF output
Choice development language: ▲ PL/M
Choice development method: ▲ MS-DOS hosted development environment with ICE etc
Price of single user development system, price per run-time (100 off): ▲ $1500, $144
Nearest perceived competitor: ▲ VRTX
Example applications: ▲ Power transmission control
Age of product, current version: ▲ 3 years, V1.3
Run-time copies and development systems in use: ▲ Not known
Distributor contact: ▲ See RMX I

RTOS-UH by IBP

(Real Time Operating System - University of Hanover)
Target processors: ▲ 68008 - 68030
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Data 8 KB. Implementation 1 KB, nucleus 8 KB, run-time 10 KB, shell 7 KB, file system 13 KB, maths 14 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ Cycle A-B-A 400 µs (68000 at 8 MHz)
Memory protection/MMU use▲ Yes
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 200 µs (68000 at 8 MHz)
Inter-task communication facilities: ▲ Semaphores, Global variables
Standard I/O drivers: ▲ RS232, Parallel printer, Floppy disk & Hard disk
Networking interfaces: ▲ Third party support for PDU-Net (Bus)
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ Proprietary RTOS-H format and MS-DOS format
User interface: ▲ Proprietary
Multi-user: ▲ Yes
High level languages: ▲ PEARL (in-house), C, FORTRAN (third party, under development)
Choice development language: ▲ PEARL (Process and Experimentation Automation Real Time Language)
Choice development method: ▲ Directly on Target
Price of single user development system, price per run-time (100 off): ▲ 300DM, 150DM (German Marks)
Nearest perceived competitor: ▲ OS-9/68000
Example applications: ▲ Controllers for pressure decasting machine, measuring forces on turning wheels
Age of product, current version▲ 5 years old, V2.2
Run-time copies and development systems in use: ▲ 3000, 6000
Distributor contact: ▲ Mr Ingo Barkus, IBP Electronic GmbH, Lilienthalstr 13, D3000 Hanover 1, West Germany. Tel: (01049) 511 630963

SMX by Micro Digital Inc

(Simple Multi-tasking Executive)
Target processors: ▲ 8086-80386
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 4.2 KB to 18.1 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 70 µs (80C188 at 10 MHz)
Memory protection/MMU use▲ No
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 317 µs (80C188 at 10 MHz) to start a task
Inter-task communication facilities: ▲ Messages, semaphores, pipes
Standard I/O drivers: ▲ Tick handler
Networking interfaces: ▲ No
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ Kernel only, but can be used with MS-DOS
User interface: ▲ No
High level languages: ▲ Microsoft C, Turbo C
Choice development language: ▲ C
Choice development method: ▲ Most debugging can be done on MS-DOS host. ICE is recommended for target debugging
Price of single user development system, price per run-time (100 off): ▲ $1495, run-time is free (no royalties)
Nearest perceived competitor: ▲ AMX
Example applications: ▲ Airline ticket printing machine, monitoring smoke-stack emissions, motor control
Age of product, current version: ▲ 1 year, V1.1
Development systems in use: ▲ 40
Distributor contact: ▲ Ralph Moores, Micro Digital, 6402 Tulagi Street, Cypress, Calif 90630, USA

Notes: ▲ *Easy to use. Supplied in linkable library form. Very low interrupt latency. Evaluation kit available. Supports development with standard MS-DOS C compilers*

Venix by VenturCom Inc

Target processors: ▲ 8086, 80286, 80386
ROMable: ▲ No, under development for Q1 1990
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 100-200 KB, file system 40 KB, drivers 100-300 KB, whole system 500 KB - 1 MB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 290 µs (80386 at 25 MHz)
Memory protection/MMU use▲ Yes
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 50 µs (80386 at 33 MHz)
Inter-task communication facilities: ▲ Messages, semaphores, shared memory, pipes, signals
Standard I/O drivers: ▲ Yes, floppy disk (ESDI, SCSI, ST506) tape
Networking interfaces: ▲ TCP/IP, Ethernet, SNA, X.25, token ring
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ Compatible with MS-DOS, NFS
User interface: ▲ Proprietary, UNIX based
Multi-user: ▲ Yes
High level languages: ▲ C (in-house) FORTRAN, PL/1, COBOL, Module-2, C++ etc
Choice development language: ▲ C
Choice development method: ▲ Directly on target hardware
Price of single user development system, price per run-time (100 off): ▲ $895, $748
Nearest perceived competitor: ▲ QNX
Example applications: ▲ Chemical process control systems, data acquisition
Age of product, current version: ▲ 8 years, V3.2
Run-time copies and development systems in use: ▲ 15000, 1000
Distributor contact: ▲ Ash Barhma, VenturCom, 215 First Street, Cambridge, MA 02142, USA. Tel: 010 617/661-1230

Notes: ▲ *Manufacturer claims that it is the only real time OS that is based on true UNIX system V. The OS was derived from AT&T source code. Hence compatibility with UNIX is assured.*

VMEexec by Motorola Computer Systems

Target processors: ▲ 68010-68030
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 13 KB

(single-processor), 15 KB (multi-processor)
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 19 µs (68020 at 25 MHz)
Memory protection/MMU use ▲ Yes, uses MMU
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 8 µs (68020 at 16.67 MHz)
Inter-task communication facilities: ▲ Message queues, semaphores, signals, events
Standard I/O drivers: ▲ RS232, comms controller, floppy disk, hard disk, SCSI
Networking interfaces: ▲ TCP/IP, DECnet, NFS
Standard real time s/w interfaces: ▲ RTEID/ORKID
Disk file system: ▲ UNIX V compatible
User interface: ▲ UNIX Bourne, C and Korne shells
Multi-user: ▲ Yes
High level languages: ▲ C, 'COFF'
Choice development language: ▲ C
Choice development method: ▲ Host with UNIX V, Remote debug with XRAY-68K
Price of single user development system, price per run-time (100 off): ▲ $5500, $110
Nearest perceived competitor: ▲ VRTX, VxWORKS
Example applications: ▲ Satellite to ground comms controller, robot-cell controller
Age of product, current version: ▲ 8 months, V1.1
Run-time copies and development systems in use: ▲ Not known, 250
Distributor contact: ▲ Motorola Computer Systems, 27 Market St, Maidenhead, Berks SL6 8AE. Tel 0628 39121

Notes:▲ Kernel based on pSOS+.

VRTX32 by Ready Systems

Target processors: ▲ 68000-68030, 68332, Philips 68070, 88000, 8086-80386, AMD 29000, NSC 32000, Z80. Planned: SPARC, TRON, MIPS, 80960.
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 8 KB, File management 64 KB, Multi-processor 14.5 KB, Networking 12 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 80 µs (68020 at 16.67 MHz)
Memory protection/MMU use▲ Yes, 'guaranteed memory protection'. MMU support given via hooks into OS
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 10 µs (68020 at 25 MHz)
Inter-task communication facilities: ▲ Mailbox, Queue, semaphore, event flag, Channels (multiprocessor), sockets (networked)
Standard I/O drivers: ▲ Defaults as provided on the Motorola 133 and 147 boards
Networking interfaces: ▲ TCP/IP
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ MS-DOS compatible
User interface: ▲ No
Multi-user: ▲ No
High level languages: ▲ Ada, C, Pascal
Choice development language: ▲ None
Choice development method: ▲ VRTX32 Velocity package, hosted on Sun 3 machine
Price of single user development system, price per run-time (100 off): ▲ £2500, £100
Nearest perceived competitor: ▲ VxWorks
Example applications: ▲ Airline flight control and instrument control; pipeline control
Age of product, current version: ▲ 8 years, V1.07.
Run-time copies and development systems in use: ▲ Not known, 'more copies than UNIX'
Distributor contact: ▲ Mr Steve Jamieson, 33 Queen St, Maidenhead. 0628 773100

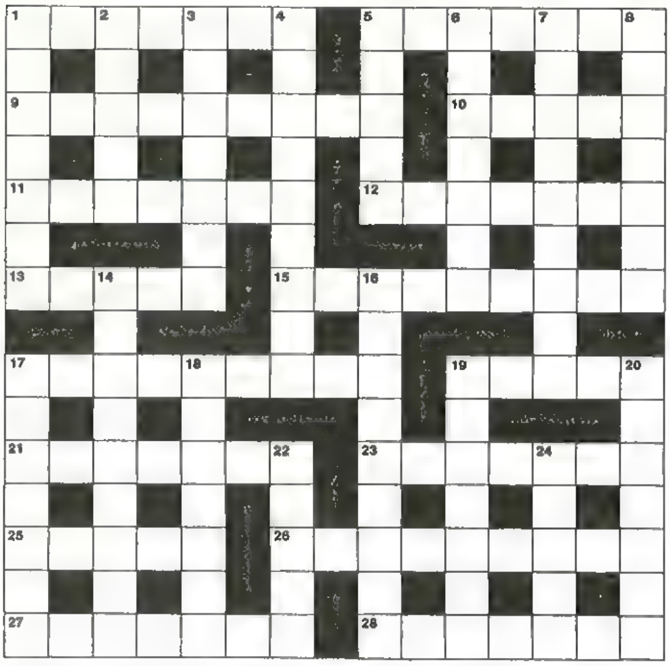
Notes:▲ VRTX32 is fully pre-emptive and priority driven. Performance is fixed and independent of system load. ARTX, its sister OS, provides Ada support including decoupled inter-task communication (as prescribed in the Ada Language Reference Manual).

VxWORKS by Wind River Systems

Target processors: ▲ 68020-68030, I960 Heurikon, Sun SPARC
ROMable: ▲ Yes
Supports re-entrant/position independent code: ▲ Yes
Memory requirements: ▲ Kernel 4 KB, full system 350 KB
Pre-emptive multi-tasking: ▲ Yes
Context switch: ▲ 35 µs (68020 at 20 MHz)
Memory protection/MMU use▲ Not known
Prioritised scheduling: ▲ Yes
Clock jobs: ▲ Yes
Interrupt handler interface: ▲ Yes
Interrupt latency: ▲ 10 µs (68020 at 20 MHz)
Inter-task communication facilities: ▲ Messages, mailbox, TCP/IP
Standard I/O drivers: ▲ Board-specific, Ethernet, serial I/O, timers etc
Networking interfaces: ▲ TCP/IP, NFS, UDP, SLIP, PRONET
Standard real time s/w interfaces: ▲ No
Disk file system: ▲ Yes, interchange with NFS, UNIX, VMS, SUN OS, MS-DOS under development
User interface: ▲ Shell C-expression interpreter
Multi-user: ▲ Yes
High level languages: ▲ C, Ada, FORTRAN
Choice development language: ▲ C
Choice development method: ▲ Symbolic & source level debugging across network on target
Price of single user development system, price per run-time (100 off): ▲ £13,500, £125 (non-networked) £250 (networked)
Nearest perceived competitor: ▲ VMEexec
Example applications: ▲ Robotics, telescope, Mars Rover project, automatic test equipment, simulation
Age of product, current version: ▲ 3 years, V4.01
Run-time copies and development systems in use: ▲ Not known
Distributor contact: ▲ Jeremy Holt, GMT Electronic Systems, Unit 7, Mole Business Park, Leatherhead, Surrey. Tel: 0372 373603. For SPARC-based system contact: Mike Carter, Mizar Europe Ltd, 0296 435214

Notes: ▲Libraries compatible with UNIX interface. Low interrupt latency.

.EXEWORD NO.2



ACROSS

1. Unit of an array in chemistry? (7)
5. Relation to George (7)
9. 0110 or thus (9)
10. Temporary permanent store in short (5)
11. Release the international flip flop (7)
12. Reserved item of program language (7)
13. Input data with a big key (5)
15. Midwifely (9)
17. Team leader goes round the first but may go round too far (9)
19. Dirty place with French coding art (5)
21. Way out value to put in the test list (7)
23. Hefty bit of house-keeping (7)
25. Highly competent in a small department (5)
26. Rich? Use it somehow for step-wise experience (9)
27. Morally right about being almost in 22 (7)
28. Fully takes over dismantled iron lungs (7)

DOWN

1. Carry out a file of this magazine (7)
2. Beat the world with a spreadsheet (5)
3. Makes values the same apparently round the middle (7)
4. Eighth nob comes unstuck below the waist (9)
5. Rest the final button (5)
6. Toy been able to make eight bits (3,4)
7. Go wrong every sixty minutes (3,6)
8. No unstable chip is wandering (7)
14. 5% of the century (9)
16. Methodical approach to a set of data (9)
17. Run a system in the theatre? (7)
18. Doubting Thomas (7)
19. Carrying out a search with a helicopter it seems (7)
20. What England does of all (7)
22. Such alcohol is liked by bar coders (5)
24. Restores the French brown by birth (5)

EXE

.EXEWORD NO.1

T	R	A	C	E		K	E	Y	B	O	A	R	D	S
E		R		X		E		E		P		E		A
L	I	B	R	A	R	Y		N	E	T	W	O	R	K
E		I		M						I		P		E
P	A	T	R	I	O	T		H	A	C	K	E	R	S
H		E		N		A		A		A		N		
O	U	R	S	E	L	V	E	S		L	I	S	T	S
N						E		H						K
E	A	R	T	H		R	E	I	N	S	T	A	T	E
		O		O		N		N		C		V		L
S	Q	U	A	R	E	S		G	A	R	B	A	G	E
P		T		R						I		R		T
E	D	I	T	O	R	S		B	A	M	B	I	N	O
E		N		R		U		U		P		C		N
D	R	E	S	S	I	N	G	S		S	P	E	C	S

.EXE REAL TIME SUPPLEMENT

STOB REAL TIME - How green is my burger

You may recall how the Observer newspaper, in a breath-takingly smug TV commercial, boasted that it had had an environmental correspondent for 20 years. Our magazine, on the other hand, has had one for just three weeks. Guess who she is.

On his way home he fancies a bite to eat, so he pops into the newly-opened, purpose-built burger joint. Inside, it is hot and bright and busy. The Lonely Guys are there, propped up uncomfortably on the ski-slope seats; reading their newspapers and spinning out the last three chips in the corner of the bag as long as they dare. Gaggles of US Burger Corps™ staff, smart as snuff in their US Burger Corps™ uniforms, are pestering their customers: sweeping the floor at their feet, wiping the clean table, snatching away food cartons as soon as they are empty; do anything, but make them go, NOW.

He walks past the rows of tubular metal tables, towards the main counter at the back of the shop. He joins the line where a young Chinese girl (dog-tired, but still very pretty) is serving. However, the adjacent queue suddenly clears and its minder, whose plastic US Burger Corps™ badge declares him to be 'Winsto', calls out, 'Yes sir! Can I help you, sir?'

Reluctantly, he gives his order: one of those big burgers and a small bag of chips, please. Winsto translates into the jargon as he works his EPOS till - one Sgt Burger Double He-Man™, one regular fries - but he forgets to say, 'And a drink with your meal?' We must fear that Winsto will not go far with US Burger Corps™. His badge incorporates a merit rating, consisting of clip-on pieces of plastic which build up into a picture of a Sgt Burger Double He-Man™. Winsto has yet to earn his first lettuce leaf.

The till transmits the transaction - RS422 differential circuit, because the length of run is greater than 50 feet - to a PC at the back of the shop. This PC is just autodialling the US Burger Corps™ mainframe with its stock order for the week. The new transaction causes it to up its order by one box...

The rack is empty of Sgt Burger Double He-Man™ boxes, so he has to wait. The people behind him get served and Winsto gets shouted at by the supervisor (whose neck and face are bright red from the heat of the Burgermatic burger cooking machine): 'Come on! Get those lines moving! Come on! COME ON!'

...spells bad news for beef cow number 47132/D, who is standing, waiting to be slaughtered, to meet the surge in demand predicted by the computer (simple stochastic model). The sound of the hydraulic, servo-controlled piston entering 47132/D's skull is familiar...

He watches while a small child pokes its finger through the crust of an apple pie. The filling splurges out, boiling hot, and the child screams. All around the walls there are sepia photographs of the Good Old Days (Chelmouth Boy Scouts Troop, 1913; His Worship the Mayor opens the Swimming Baths, 1933), interspersed with the US Burger Corps™ corporate logo: a dancing cow munching a Sgt Burger Double He-Man™. He suffers a sudden twinge of nausea.

...down from the satellite link to the SQL database. A man in a suit turns from the VDU and says, in Spanish, 'They will take as much as we can send them. Get the chain-saws oiled up - we start tomorrow.' Outside, in the forest, the last example of a species of herb that could cure cancer enjoys its final few hours of anonymous existence...

'I've changed my mind. I want to cancel my order.' Winsto favours him with a disdainful glance. 'I'm sorry sir, that's not possible. You're too late.'

And so he was.

PSYCHOMETRIC TESTING METHODS

The way to find the best candidates would surely be to test them. Unfortunately, it can also unnerve them.

Most people who apply for a job in the next decade will be asked to submit to some form of psychological or psychometric test. Knowing what they are intended to reveal about you, you should not only allay any fears, but make it an opportunity to enjoy thinking about nothing but yourself for an hour or two.

There are three categories of psychological tests. Tests published by popular magazines for women are intended for people who, like most of us, enjoy a bit of harmless self indulgence. There is nothing wrong with them provided that they are taken as seriously as the computer games disk. The scoring mechanism of these questionnaires reflects the level of scientific data used to compile them - not much.

Full psychological tests which are approved by the British Psychological Society or other professional institutions are at the opposite end of the spectrum. These are scientific instruments developed from years of research and are usually only administered by an industrial or occupational psychologist. If you are asked for a battery of tests such as this and have any doubts, you could ask the following questions:

- *'Is this method approved by The British Psychological Society?'*
- *'How were the results validated?'*
- *'Who else uses this system?'*
- *'How long has this method been around - has it stood the test of time?'*

If you are satisfied with the answers, then the best attitude to take is to relax and look on it as you would if you have to have your brain tested by an EEC scan. You are with professionals who are intending to find out your strengths and weaknesses and you know the answers already - you have nothing to be worried about at all.

Psychometric testing methods used most widely in industry are those that are developed to be used by lay people after a minimum of training by the manufacturer of the system. Industry wants something that works, that improves selection methods and helps to motivate employees to improve their performance in the workplace and they are not concerned with knowing every facet of a person's personality. The psychometric testing systems are the ideal tool.

We prefer to show the results to a candidate and discuss the findings as part of an interview, but some companies think it is not a good idea as the results are displayed on non-linear graphs which make sense to readers of this Magazine, but not to the non-numerate brutal and licentious members of society.

DISC, the system we use, isolates four measurable behavioural characteristics that are needed in the workplace. Each of us display some of them at times, but the criteria of DISC is which type of behaviour is most like you and which is least like you. When these measurements are plotted on to three graphs, you get a picture of the person's self-image and how they modify that behaviour pattern under pressure.

The big surprise was that, despite a very large number of permutations which could appear, people fell into classic patterns, behavioural blueprints. This seems more natural when you consider the biological basis of man; dogs at work tend to be either gun dogs, cattle dogs or hounds.

Judy Hortin, Recruitment Consultant, Acumen Search & Selection

SOFTWARE QUALITY CONTROL ENGINEER
to £16,000 + BENEFITS

QUALITY – GETTING IT "RIGHT FIRST TIME, EVERYTIME."

At LBMS, quality is our hallmark. Our investment in quality is based on the firm belief that it will be repaid in full through total customer satisfaction.

We are one of the nation's leading IT consultancies, providing an integrated package of services including training, consultancy and Computer Aided Systems Engineering (CASE) tools to a broad client base spanning leading "blue-chip" multi-nationals and Government organisations. We were co-developers of SSADM and are presently leading the CASE market with our AUTO-MATE PLUS product.

Considerable business growth has created a new opportunity for a Software Quality Control Engineer to be involved with the forthcoming release of new Windows-based software product. Working in a small team of Quality Engineers, you will liaise closely with development and customer support staff on functional and system level testing, destructive testing, test plan generation and product releases. Due to a rapidly evolving Quality Management programme, you may also expect to attend Fagan Inspections and contribute to new quality procedures.

From a software development or PC environment, you will have at least 12 months QC experience and an understanding of MS-DOS. You must be a good communicator with a strong commitment to quality and a knowledge of BS5750/ISO9000.

The salary of up to £16,000 will be supported by excellent benefits. Career prospects within a quality organisation where promotion is based purely on merit, are excellent.

Interested? Call David Drysdale on 01-636 4213. Alternatively write with your career and salary details to him at Learmonth & Burchett Management Systems Plc, Evelyn House, 62 Oxford Street, London W1N 9LF.

LBMS

WEST YORKSHIRE

Programmers (COBOL, 'C', BASIC and 4GLs), Analyst Programmers, Software Engineers, Project Leaders, Support (pre and post sales), Trainers (MSDOS, UNIX and Applications).

FOR YOUR NEXT CAREER MOVE AROUND WEST YORKSHIRE:

Telephone Vincent Atherton on Leeds (0532) 504560 or write to:

AIREDALE RECRUITMENT
Realtex House. Micklefield Lane.
Rawdon, Leeds LS19 6AX.

AIREDALE RECRUITMENT

WANTED! 'C' MEN & WOMEN REWARDS;

LONDON **£20k-30k**
System Designers required to design top-level specification for state-of-the-art **DATABASE** product including Superior Screen handling facilities. Knowledge of **MS-DOS, 'C'**, with either **OS/2** or **UNIX**. Familiarity with databases such as dBase or Informix useful.

CAMBS **to £20k plus bonus**
Graphical Display Systems Developer seeking degree level candidates with experience of either **'C', PASCAL** or **FORTRAN** on **VMS, UNIX** or **PRIMOS**. Pension, flexitime and relocation.

BERKS **£16k-£25k**
Developer of secure office automation systems seeking experience in one of the following: **'C', SOFTWARE PORTING, LAN/WAN, ISO/OSI, X25, TOKEN RING, UNIX** in a Development, Support, Analysis or Networking role.

E. MIDLANDS **£14k-£20k**
Small is Beautiful! Designers/Programmers required for DataComms developments. Experience of two of the following is required: **'C', X.25, ASSEMBLER, WINDOWS, SQL, UNIX** or **NETWORKING** - training given in those areas which you lack.

W. COUNTRY **£Neg.**
Become involved in the development of Process Control Systems across a wide spectrum of hardware, communications and systems. Experience necessary with any three of: **PASCAL, BASIC, MS-DOS, PLC's** or **VAX/VMS.**

LONDON **c£15k Neg.**
Analyst/programmer and technical support person sought by leading independent book publisher. Knowledge of either **MS-DOS** or **VMS** required with one of the following: **'C', SQL FORMS, PRO C** or **ORACLE** (training provided).

OXON **to £21k**
Strong believers in the excellence of technology and design for **PC WORKSTATIONS COMMUNICATIONS, VIDEO DISC, CD-ROM, TOUCH SCREEN** and **GRAPHICS**, our client is keen to recruit people with good 'C' experience.

E. ANGLIA **£13k-£18k**
Programming for development and support of **APPLE MAC** and PCs running **PRESENTATION MANAGER** and **WINDOWS**, including PC based **DATABASES** and **INTERNALS**. Experience of **'C', C++** or **PASCAL** on PCs essential.

LONDON **to £20k**
Real-time development of **VAX/VMS** and **TRANSPUTER** software, Knowledge of **'C'** and **OCCAM** essential. Fortran, parallel algorithms, methods or Apple Mac useful.

MANY MORE INTERESTING POSITIONS THROUGHOUT THE U.K!

Contact to find out more about these posts or the many others that we have. As specialists in the industry we can provide you with a professional and confidential service in assisting you with the right career move. Telephone or write to either Antony Bridge M.A., Judy Hortin or Ed Butcher.

01-392 1514

ACUMEN
SEARCH & SELECTION INTERNATIONAL
No 7 Tideway Yard, Mortlake High Street, London SW14 8SN
Tel 44 (0)1 392 1514 Fax 44 (0)1 392 1518

Evenings till 9pm 01-878 8206

ADVERTISER INDEX

STOB at the Show

Stob, a programmer by trade, barbours ambitions of becoming a full time reporter. To help her gain experience, we sent her along to cover the Computer Computer Show.

O frabjous day! Callooh! Callay! It's time once more for the Computer Computer Show; held, as ever, in the National Carbuncle Conference Complex, Birmingham. I travelled up by the special Show Train, laid on for trade journalists by the well-known British printer manufacturer Nippo-Epcot. As we left Euston at the crack of 9:20 am, I wandered around our first class compartment, getting to know my fellow yawning journos. One question which seemed to be on everybody's lips – lips somewhat stained by the complimentary Full English Breakfast with Bucks Fizz – was 'Why can't they hold it in London, like the other shows?'

No matter. The train finally pulled in at Birmingham Carbuncle, half an hour late (having stopped in a field for that period of time, to help promote the atmosphere of relaxed contentment portrayed in BR's award-winning TV advertising campaign). Clutching free copies of the best-selling paperback *1001 amusing things to do with a Nippo-Epcot dot matrix printer*, the press corps disembarked and headed for the exhibition hall.

On the IBM stand, they were showing off a host of new SAA-compliant products. SAA has been extended beyond mere software. Next year's aware corporate executive will have a CUA-conformant pencil holder and flip-over calendar next to his PS/2. But don't go thinking the additions stop at desk furniture. IBM has also published a list of SAA-conformant forenames. No space to repeat them all here; I can tell you that Lindsay, Robert and William are allowed, Deirdre is dodgy but John, Jack and Verity are out. An IBM spokesman said, 'Obviously we do not expect everybody to change his name at once. However, we do anticipate this scheme being phased in within the next three months at sites using 370-architecture equipment.'

Enough of Big Blue. Following the runaway success of *Dentist and Volvo Driver Magazine*, EPAP-GNU Publishers has launched two new combined trade-and-hobby titles, half-aimed at the computer market. *Systems Analyst and Wooden Toy Maker* and *Programmer and Train Spotter* subscriptions were both available free at the EPAP-GNU stand, provided that you signed a little card stating that you were the DP Manager of a giant multinational, with a computer hardware budget exceeding £2.5 million.

Hardware product of the show was Watchix, the first wrist computer to run UNIX System V. Watchix continues the trend of miniaturised personal computers with an 80 x 25 character LCD screen that is quite unreadable and a 101 key keyboard too small for a gnat's fingers.

On the software front there was FRONART, the FORTRAN rescrambler. This accepts beautifully-structured, commented FORTRAN-77 code and outputs it as raw spaghetti FORTRAN IV, riddled with old-fashioned DO loops, GOTOs galore and, to add a little spice, the occasional introduced bug. The product is aimed at the subcontractor market, where the need is to keep the price of code maintenance high. It is selling very well, and FRONART's developers are now working on a follow-up – a C-to-BASIC translator.

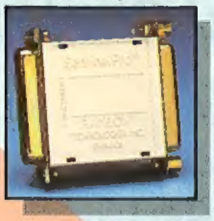
But already, sadly, it's time to go. As the train slips out, Metropolis-bound, accompanied by the merry carolling of the press rendering of 'Every Sperm is Sacred', we must say goodbye and God Speed! to the Computer Computer Show '89. EXE

S O F T W A R E   S E C U R I T Y

With £1 Billion Worth Of Protected Software...

Rainbow Is The Safe Bet.

SentinelPro™



• Runs under DOS, OS/2 and Xenix • Algorithm technique (Never a fixed response) • External parallel port installation • Minimal implementation effort • Higher level language interfaces included • 100 times faster than fixed-response devices (1 ms) • ASIC design for reliability

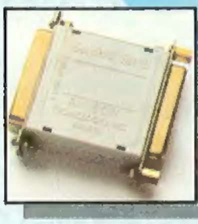
Sentinel-C™

• Protects multiple packages with one device • 126 bytes of non-volatile memory programmed before shipment of the software • Rainbow supplies a unique adapter for programming the unit • Higher level language interfaces included • Runs under DOS, OS/2 and Xenix • External parallel port installation

Eve™

• For the Macintosh SE and II • Complies with Apple Desktop Bus Interface requirements • Rainbow-assigned developer passwords to prevent tampering by other developers or sophisticated "hackers" • 7 locks per key, usable individually or in combination, on one or up to seven applications

SentinelShell™



• Runs under DOS on IBM PCs and compatibles • Protects without requiring access to the source code • Completely transparent to the end user • User-friendly software • Pocket-size key attaches quickly to any standard PC parallel port • ASIC design for reliability

DataSentry™

• Completely user-installable • Pocket-sized external device • Menu-driven, user-friendly interface • Single- or multi-user security system • Audit trail, log-on identifiers and automatic encryption/decryption of entire directories • Secures data transmitted by modems • Prevents recovery of data by utility programs

Software developers creating the latest applications for the IBM PC/XT/AT, PS/2 and compatible systems can now turn to the Software Sentinel range of hardware keys for the *first-class* in *world-class* software protection.

There's the best-selling SentinelPro, known worldwide for its virtually unbreakable security, its ASIC technology and its invisible operation.

A close relation, the Sentinel-C for custom configurations, enables multiple package protection with a single device.

For the Apple market, security-minded Mac software developers can now secure their return-on-investment, too. Eve plugs into the Mac ADB connector and is completely transparent to the user—providing up to seven programmable security locks per key.

Rainbow's latest protection strategy is the SentinelShell—that lets users place a "shell" around existing, off-the-shelf programs. Access can be limited to those issued a key, so that libraries, educational establishments and corporations can very simply guard their investments.

Available soon from Rainbow is the DataSentry, a user-installed key that provides low cost security for sensitive data in both database applications and corporate/banking environments.

No matter where you sell your software worldwide, stay in control of your distribution and revenue by choosing the internationally accepted standard in protection... Rainbow Technologies. Be sure. Protect your pot of gold at the end of the rainbow.

CALL TODAY OR ORDER NOW.

☐ Please send me a SentinelPro Evaluation Kit.
I enclose £50 + VAT Payable to Rainbow Technologies Ltd.
☐ Please debit my credit card. Access ___ Visa ___ Amex ___
Expiry Date ________ (PLEASE TICK)

☐ Please send me more information.

Name ____________________

Position ____________________

Address ____________________

Telephone ____________________

Signature ____________________

RAINBOW TECHNOLOGIES LTD., Shirley Lodge, 470 London Road, Slough, Berkshire SL3 8QY.

RAINBOW TECHNOLOGIES

Shirley Lodge, 470 London Road
Slough, Berkshire SL3 8QY
Tel: 0753-41512 Fax: 0753-43610

CIRCLE NO. 736